Universidad Zaragoza

Trabajo Fin de Grado
Grado en Ingeniería Informática

Desarrollo de mecanismos de publicación de datos para el estudio de la calidad del aire en entornos urbanos

Autor
Héctor Ochoa Ortiz

Director
Javier Nogueras Iso

Escuela de Ingeniería y Arquitectura
Curso 2019-2020

Resumen

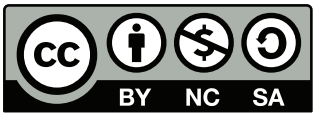
Este TFG se ha realizado dentro del contexto del proyecto TRAFAIR (Understanding traffic flows to improve air quality), un proyecto europeo que pretende por un lado monitorizar la calidad del aire en contextos urbanos, y por otro lado predecir esa calidad en base a flujos de tráfico y previsiones meteorológicas.

El objetivo del TFG es proporcionar los mecanismos de publicación de datos requeridos tanto internamente para acceder a los datos necesarios para la ejecución de los modelos de predicción, como externamente para la difusión de los resultados de la predicción y monitorización.

De cara a la publicación interna se ha desarrollado una aplicación Web que permite acceder a datos de meteorología, red viaria y factores de emisión. Dado el carácter geográfico de los mismos, se han utilizado Sistemas de Información Geográfica (SIG) para su modelado y almacenamiento en una base de datos espacial.

De cara a la publicación externa se han integrado y extendido tecnologías que permiten publicar los datos tanto dentro de Infraestructuras de Datos Espaciales (IDE) como en portales de Datos Abiertos.

Se ha desarrollado software para la generación automática de metadatos a partir de la información de conjuntos de datos publicados en Geoserver mediante servicios OGC como metadatos GeoDCAT-AP en servidores CKAN. Siendo esta solución reutilizable en cualquier proyecto de información geográfica.



Agradecimientos

A mis compañeros de Mapeado Colaborativo, que tanto me han ayudado a crecer en el mundo de los SIG.

A mis amigos, por hacer más llevadero el periodo de confinamiento.

A mi madre, Mercedes, por apoyarme en la carrera universitaria y en mi crecimiento como persona.

Y al director del trabajo y mi tutor en el proyecto, Javier Nogueras, y a la líder del proyecto, Raquel Trillo, por darme la oportunidad de entrar en el proyecto TRAFAIR y aguantarme durante este tiempo.

Este TFG se ha realizado dentro del contexto del proyecto europeo TRAFAIR, el cual ha sido cofinanciado por el programa Connecting Europe Facility de la Unión Europea (número de referencia de proyecto: 2017-EU-IA-0167).

Índice general

1 | Introducción

1.1. Contexto y motivación

Este Trabajo de Fin de Grado (TFG) se enmarca dentro del proyecto europeo TRAFAIR (*Understanding traffic flows to improve air quality*)[1], financiado por el programa Connecting Europe Facility de la Unión Europea (Proyecto Nº 2017-EU-IA-0167). El consorcio que desarrolla este proyecto reúne a 10 organizaciones de Italia, Bélgica y España, entre ellas la Universidad de Zaragoza, para desarrollar servicios innovadores empleando datos de calidad del aire, condiciones meteorológicas y flujos de tráfico, en beneficio tanto de los ciudadanos, como de las autoridades responsables de la toma de decisiones que afectan a la calidad del aire.

Los principales objetivos del proyecto son la monitorización en tiempo real de la contaminación del aire en zonas urbanas, y el desarrollo de un servicio de predicción de la calidad del aire basado en predicciones meteorológicas y del flujo de tráfico urbano. Los resultados del proyecto se validan en seis ciudades piloto (Módena, Florencia, Pisa, Livorno, Santiago de Compostela y Zaragoza) y en cada una de ellas se pretende desarrollar un modelo del tráfico rodado para que, junto con la predicción de las condiciones meteorológicas, principalmente datos de la dirección y velocidad del viento, se analice la dispersión de los contaminantes dentro de la ciudad. Por otro lado, también se instalan una serie de sensores electroquímicos de bajo coste para medir niveles de contaminación repartidos en distintos puntos de cada ciudad. Además, se pretende publicar los datos de monitorización y predicción de calidad del aire como datos abiertos y crear diferentes aplicaciones móviles para informar a los ciudadanos de la calidad del aire que respiramos y tratar de concienciar en un uso responsable del automóvil privado. En la figura 1.1 se resumen los principales datos de entrada y objetivos del proyecto [2].

En este TFG se pretende contribuir a los desarrollos implicados en los procesos necesarios para la publicación interna y externa de datos. Por publicación interna nos referimos al procesamiento y generación de los datos

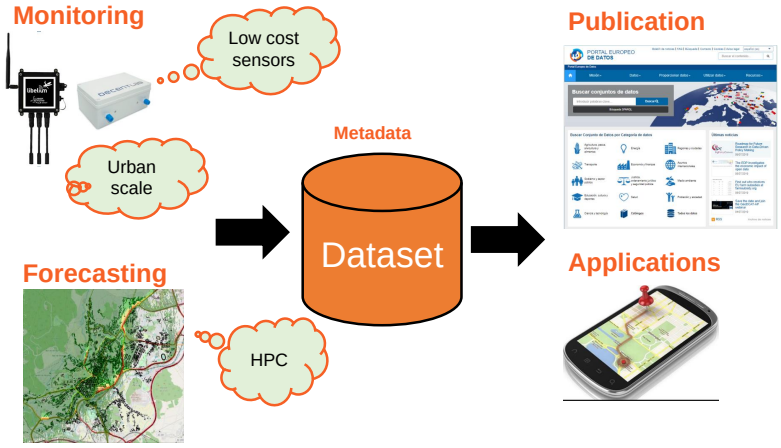


Figura 1.1: Principales datos de entrada y objetivos de TRAFAIR

que son necesarios como entrada para los modelos de predicción. Por publicación externa nos referimos al procesamiento que es necesario realizar para incluir los conjuntos de datos de monitorización y predicción que se generan en TRAFAIR dentro de iniciativas de portales de datos abiertos.

Como datos abiertos nos referimos a aquellos que tienen una licencia que permite su reutilización sin ningún permiso específico, y son accesibles para el público general[3].

1.2. Objetivos

El objetivo de este TFG es proporcionar los mecanismos de publicación de datos requeridos tanto internamente para acceder a los datos necesarios para la ejecución de los modelos de predicción, como externamente para la difusión de los resultados de predicción y monitorización. Dado el caracter geográfico de los datos utilizados y generados en el proyecto TRAFAIR, para el modelado, almacenamiento y gestión de los datos se utilizan las herramientas habituales de los Sistemas de Información Geográfica (SIG) y las Infraestructuras de Datos Espaciales (IDE)[1].

Respecto a los mecanismos de publicación, este TFG cuenta con dos secciones claramente diferenciadas: la publicación interna de datos, donde se transforman diversos conjuntos de datos de la base de datos en fases tempranas del proyecto y se vuelcan por una API para su uso por parte de los investigadores, tanto para corroborar la corrección de los datos como para

[1]Una IDE se define como un conjunto de tecnologías, políticas y disposiciones institucionales que facilitan el acceso y explotación de datos espaciales[4].

servirlos de entrada a la simulación en las fases medias del proyecto; y la publicación de datos abiertos, donde se seleccionan los conjuntos de datos con relevancia para el público general y se ponen a la disposición de la ciudadanía en diversos formatos y siguiendo los estándares descritos en este documento.

También se pretende que los datos abiertos sean interoperables entre administraciones usando los estándares y normas internacionales. En primer lugar, el proyecto persigue que los conjuntos de datos sean accessibles a través de los servicios habituales de localización o catálogos de las IDE. Es decir, queremos que los metadatos que describen a los conjuntos de datos geográficos generados en TRAFAIR sean localizables a través de servicios de catálogo de una IDE que a su vez son unos servicios conformes a la especificación Catalog Services for the Web (CSW)[5] propuesta por el consorcio Open Geospatial Consortium (OGC).

Pero como los servicios de catálogo de una IDE no son fácilmente accesibles desde dominios más generales, también se pretende integrar los metadatos que describen los conjuntos de datos (y los servicios de visualización y descarga) de TRAFAIR dentro de las iniciativas de portales de datos abiertos, liberándose a toda la ciudadanía mediante su publicación en un portal de datos abiertos propio del proyecto además de estar incluidos los diferentes conjuntos en los diversos portales de las administraciones: desde portales locales, a portales nacionales, o el portal de datos Europeo.

1.3. Tecnologías utilizadas

Se ha decidido utilizar Python3[6] como lenguaje principal del proyecto ya que es el lenguaje con el que el autor del proyecto se siente más cómodo. Al tener una baja carga de trabajo las pasarelas que se han tenido que programar, ser la temporización del proyecto no crítica y ser la comunicación con los diferentes servicios por medio de APIs y peticiones HTTP, se puede utilizar una gran variedad de lenguajes sin influir en el resultado final. Se ha seguido la guía de estilo PEP8[7] de normalización de Python.

Para facilitar el acceso a los datos del proyecto a través de de las especificaciones estándar habituales en el mundo de las IDE y propuestas por la organización Open Geospatial Consortium (OGC) se ha utilizado el software Geoserver [8]. Este software permite poner en marcha un servidor conectado con la base de datos espacial PostGIS del proyecto y ofrecer, entre otros, los siguientes servicios compatibles con las especificaciones de OGC: servicio de visualización WMS (Web Mapping Service)[9], servicio de descarga de fenómenos discretos WFS (Web Feature Service)[10] y servicio de descarga de coberturas (WCS)[11].

Como herramienta de distribución de datos abiertos en un portal se ha utilizado el software CKAN[12]. Se ha elegido por su facilidad de uso y su extensa documentación y extensiones para facilitar cumplir los objetivos del proyecto, además de ser también la herramienta utilizada en numerosos portales de referencia (Gobierno de Aragón, Gobierno de España, Comisión Europea).

Además de los servicios OGC brindados por GeoServer, se ha utilizado también el software PyCSW[13]. Este software permite poner en marcha un servidor de catálogo de metadatos geográficos compatible con la especificación de OGC para servicios de catálogo denominada Catalog Services for the Web (CSW)[5]. Este software tiene la ventaja de integrarse fácilmente con los servidores CKAN gracias a la extensión ckanext-spatial[14].

1.4. Estructura de la memoria

El resto de la memoria se compone de los siguientes capítulos:

- El Capítulo 2 proporciona una visión general del proyecto, incluyendo diagramas de arquitectura, y realiza breves anotaciones sobre el diseño.
- El Capítulo 3 expone la primera sección del trabajo: la publicación interna de datos.
- El Capítulo 4 explica la segunda sección del trabajo: la publicación de Datos Abiertos.
- El Capítulo 5 muestra la puesta en marcha y validación de ambas secciones.
- El Capítulo 6 explica cómo se ha llevado la gestión del proyecto y desarrolla las conclusiones sobre el trabajo y trabajo futuro a realizar.
- Se incluyen además dos anexos.
    - El Anexo A detalla la instalación de CKAN con sus complementos.
    - Y en el Anexo B se enseña el ejemplo de un conjunto de datos en sus diferentes fases dentro de la Publicación de Datos Abiertos.

2 | Visión general

En la figura 2.1 se puede observar la arquitectura general del proyecto TRAFAIR. Esta se compone de cuatro fases diferenciadas.

Primero se encuentra la fase de Entrada, que recoge información relevante para el proyecto de diversas fuentes de datos, propias y externas. Estos datos se transforman y se introducen en la base de datos del proyecto para su uso en fases posteriores.

En segundo lugar, tenemos la fase de Modelado/Simulación, donde se aúnan las entradas en modelos que ayudan a entender el problema planteado en el proyecto. Como objetivo de esta fase se desea llegar al modelo lagrangiano GRAL[15] y realizar con esos datos la simulación del estado de la calidad del aire en las ciudades del proyecto día a día.

Esta simulación genera una salida que se guarda en la base de datos y posteriormente en la última fase, la de Publicación de Datos Abiertos, se libera al público, junto con los datos de entrada que son recogidos directamente por el proyecto (mediciones de sensores de bajo coste).

En la figura 2.2 se muestra una arquitectura en capas donde se puede ver los principales almacenes de datos, los metadatos que los describen y los componentes software que facilitan el acceso y publicación de los mismos desde el exterior. Además, en esta figura se puede ver que el proyecto pretende ser accesible desde dos perspectivas distintas: desde el punto de vista de las Infraestructuras de Datos Espaciales (IDE), y desde el mundo de los portales de Datos Abiertos. Respecto a la publicación interna, se puede ver cómo el componente Web App permite acceder a los datos necesarios por GRAL. Por otro lado, la publicación externa se consigue a través de portales CKAN que facilitan el acceso a los conjuntos de datos abiertos y que enlazan con los contenidos reales a través de servicios del mundo IDE conformes con las especificaciones establecidas por el Open Geospatial Consortium (OGC): Web Mapping Services (WMS) para la visualización, Web Feature Services (WFS) para la descarga de datos de fenómenos discretos (objetos reconocibles que tienen un contorno o una extensión espacial bien definida a los que se asocian

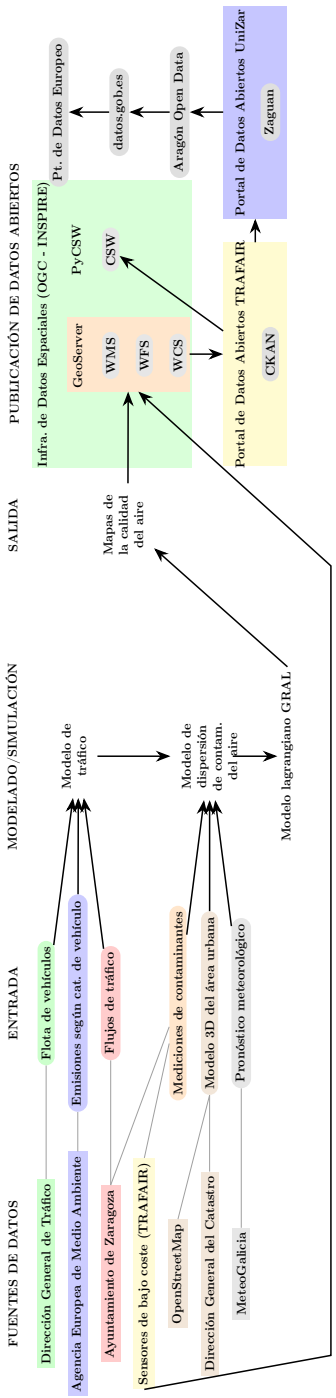


Figura 2.1: Flujo de datos del proyecto TRAFAIR dentro de la ciudad de Zaragoza

atributos temáticos, por ejemplo, edificios con altura, tipo de habitabilidad, fecha de construcción, ...), y Web Coverage Services (WCS) para la descarga de datos de coberturas (fenómenos continuos que varían sobre el espacio y sin una extensión específica, por ejemplo, una cobertura con la temperatura sobre la superficie del mar).

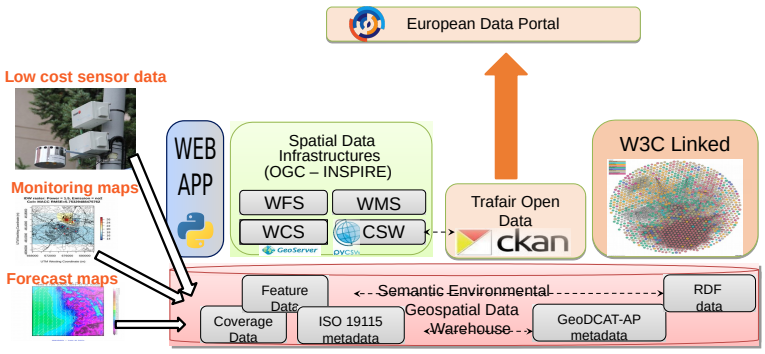


Figura 2.2: Arquitectura de componentes de datos y servicios en el proyecto TRAFAIR

2.1. Publicación interna de datos

La aplicación de publicación interna de datos se enmarca dentro de la fase Modelado/Simulación del flujo de datos del proyecto (figura 2.1). Su función es generar archivos intermedios dentro de la fase para la simulación y visualización a partir de los datos introducidos en la base de datos del proyecto.

En la figura 2.3 se muestra el flujo de datos implicado en el modelado/-simulación y los conjuntos de datos que es necesario proporcionar al modelo GRAL [16]. En primer lugar se genera un modelo de tráfico con el simulador SUMO, calculando el flujo estimado para cada calle de la ciudad. Posteriormente se calcula el modelo de emisiones con VEIN, que calcula las emisiones NOx que produce el modelo de tráfico del paso anterior, y este valor de contaminante se guarda en la base de datos asociado a cada calle. Este trabajo de simulación se desarrolla en el TFG de David Sáez [17].

La aplicación de publicación interna de datos genera entonces los archivos necesarios como entrada a GRAL a partir de la base de datos, tanto de red viaria con emisiones NOx, como de factores de emisión y predicción meteorológica. Se explica detalladamente cada función en el capítulo 3.

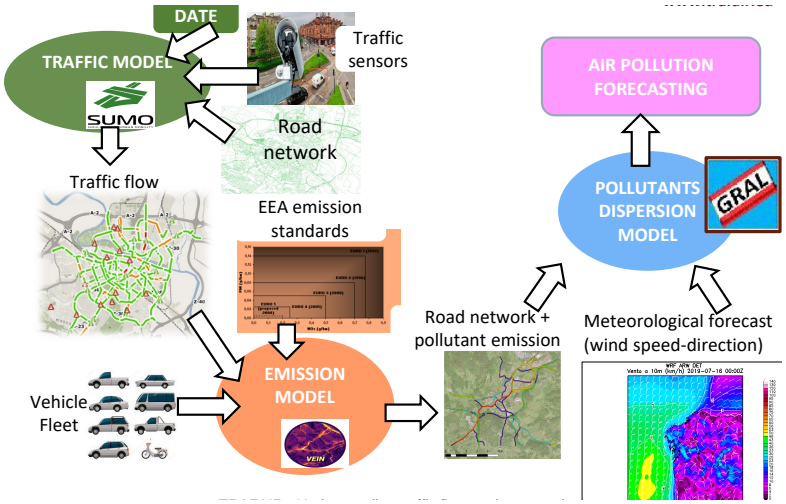


Figura 2.3: Datos de entrada necesarios para la ejecución del modelo de predicción GRAL

2.2. Publicación de datos abiertos

La aplicación de publicación de datos abiertos se localiza en la fase Publicación de Datos del diagrama de flujo de datos del proyecto (figura 2.1).

La publicación de servicios OGC únicamente no se puede considerar publicación de Datos Abiertos. Para conseguir que los datos sean realmente accesibles como Datos Abiertos, necesitamos registrar los conjuntos de datos en los portales de Datos Abiertos oficiales. Además, la publicación de conjuntos de datos en el Portal Europeo de Datos[18] es un requerimiento del proyecto.

Con el objetivo de registrar los datos de TRAFAIR como datos abiertos, el primer paso ha sido seleccionar un perfil de metadatos apropiado conforme con los modelos de metadatos aceptados en el contexto del Dominio Abierto. Teniendo en cuenta el carácter espacial de los datos generados por TRAFAIR, se ha adaptado el perfil de metadatos GeoDCAT-AP[19]. GeoDCAT-AP es un perfil de metadatos que extiende DCAT-AP[20][21], un perfil de metadatos diseñado por la Comisión Europea para describir datos del sector público. Las propiedades de metadatos GeoDCAT-AP han sido diseñadas para asegurar la conformidad con los requerimientos de metadatos de la directiva europea INSPIRE para establecer una infraestructura de información espacial en Europa [22].

Por otro lado, para minimizar el esfuerzo de crear metadatos y el registro de esos datos en portales de Datos Abiertos, hemos decidido automatizar este

proceso por medio de un software que recupera las capacidades de los servicios OGC y convierte esta información a registros de metadatos que posteriormente son insertados en un servidor de Datos Abiertos CKAN. CKAN[12] es la plataforma de código abierto más ampliamente usada para mantener portales de Datos Abiertos, que incluye los complementos necesarios para intercambiar metadatos en formato RDF (el formato de serialización utilizado para GeoDCAT-AP)[23].

Se valoró también como herramienta para gestionar un portal de datos abiertos nos encontramos con dos principales el uso de DKAN[24].

Ambos son muy similares, con la diferencia de CKAN estar programado en Python y DKAN en PHP (Drupal). Dado que CKAN tiene una comunidad mayor, y más documentación e integración con otras herramientas se escogió como software para este proyecto.

Diversos complementos se han usado para completar los requerimientos del proyecto, listados en el anexo A, donde se explica la instalación en detalle de CKAN junto con los complementos que se han utilizado. También, al desear que los datos sean accesibles dentro de una IDE, se pone en marcha un servicio CSW[5] con la conexión de PyCSW[13] con CKAN via ckanext-spatial[14].

3 | Publicación interna de datos

Dentro de las fases intermedias del proyecto (Entrada y Modelado/Simulación) se ha decidido realizar una aplicación para facilitar la extracción de datos por parte de los investigadores involucrados.

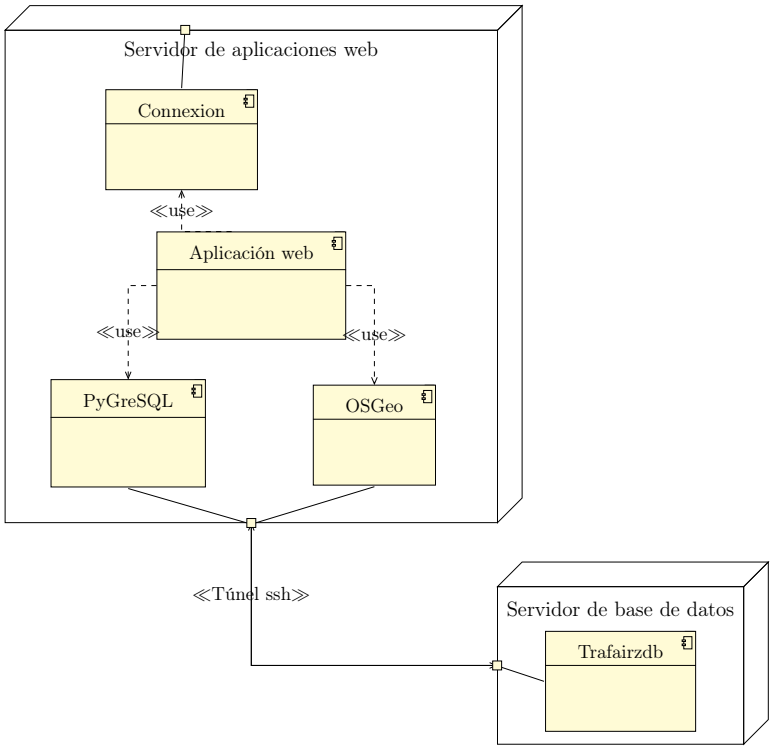


Figura 3.1: Diagrama de despliegue de la aplicación web para la publicación interna de datos.

En la figura 3.1 podemos ver los componentes que componen esta parte del proyecto, con el nodo donde se encuentra nuestra aplicación web que se conecta mediante las librerías necesarias a la base de datos, y acepta

peticiones a la API mediante el puerto abierto para la librería Connexion, explicada en la siguiente sección, además de una interfaz web para la API.

Dado que no todo el mundo en el proyecto tiene conocimientos de bases de datos, la interfaz gráfica de esta aplicación resultó muy cómoda para que los investigadores pudieran comprobar la consistencia y verificar la corrección de los datos introducidos en la base de datos, además de poder extraer transformaciones de manera automática en los formatos requeridos para lanzar las simulaciones en GRAL[15]. Los ficheros generados siguen la documentación de GRAL[25] y son explicados en las secciones posteriores.

Se han implementado las siguientes funciones en estos tres grupos.

1. *roads*, un conjunto de datos conteniendo la morfología de la red viaria en una zona urbana delimitada por un rectángulo envolvente geográfico e incorporando la estimación de emisiones de NOx. Ahí se han creado tres funciones: *descarga de fichero shapefile con nox a partir del viario en la base de datos*, *descarga de fichero line.dat a partir del viario en la base de datos* y *descarga de fichero line.dat a partir del viario en shapefile*.

2. *weather*, predicciones meteorológicas para las próximas 24-48 horas a partir de un instante temporal. Esta función incluye la descarga de los ficheros *meteopgt.all* y *mettimeseries.dat*.

3. *emissions*, con datos de factores de emisión. Esta función descarga de fichero *emissions_timeseries.txt*.

3.1. Aplicación web con Connexion

Esta aplicación se implementó con una API y una interfaz gráfica con OpenAPI[26] (anteriormente conocido como Swagger), todo realizado con la librería Connexion[27] para Python3, para transformar y devolver datos sobre la red viaria, meteorología y emisiones contaminantes.

En el diagrama de clases (figura 3.2) se puede ver la estructura del programa de publicación interna de datos. La función principal (*main*) de la clase app inicializa, configura e inicia un objeto FlaskApp, que es quien controla y gestiona la API según los datos del fichero YAML (ver código 3.1) pasado como parámetro en la función *add_api()*.

Cada vez que recibe una petición a la API busca en el fichero (campo *operationId*, ejemplo en la línea 59) a qué clase y función Python dentro del paquete tiene que llamar para realizar las gestiones deseadas, estas funciones se guardan dentro del paquete src/app.

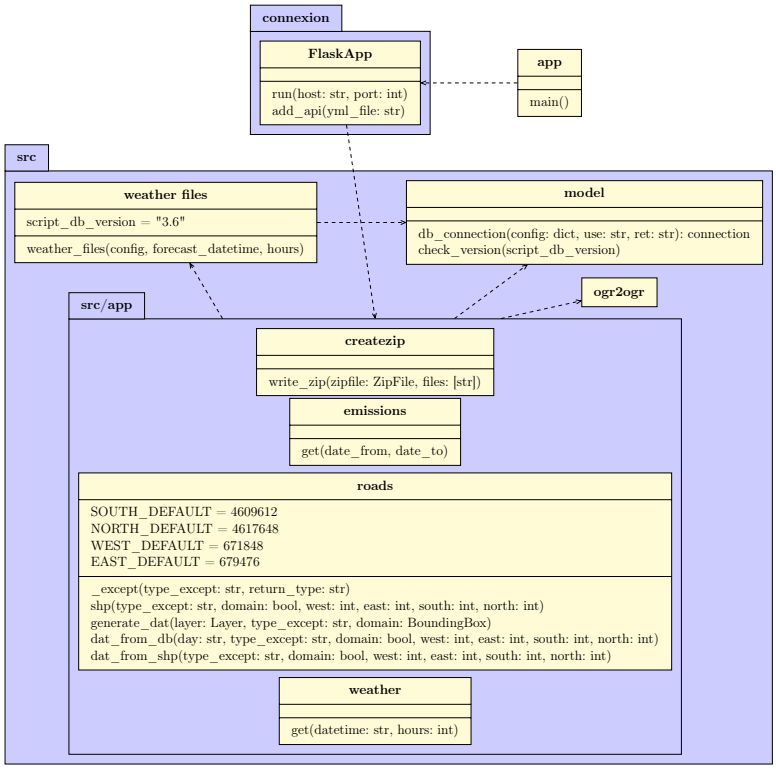


Figura 3.2: Diagrama de clases aplicación web publicación interna de datos

En las siguientes secciones se explican las distintas funciones llevadas a cabo.

```yaml
openapi: "3.0.0"
info:
    title: "TRAFAIR DATA EXTRACTING"
    version: "1.0"
tags:
    - name: "weather"
      description: "Datos meteorol\u00f3gicos"
    - name: "roads"
      description: "Red viaria"
paths:
    /roads/shapefile:
        get:
            parameters:
              - in: query
                name: type_except
                schema:
                  type: string
                description: "Tipos de vía a eliminar, separados por \",\" (sin espacios). Ejemplo: track,service"
                required: false
              - in: query
                name: domain
                schema:
                  type: boolean
                description: "True si se desea usar un rectángulo para limitar el dominio de la salida, False en caso contrario (por defecto True)"
                required: false
              - in: query
                name: west
                schema:
                  type: number
                description: "Límite oeste del dominio en coordenadas proyectadas (por defecto 671848 si domain es True)"
                required: false
              - in: query
                name: east
                schema:
                  type: number
                description: "Límite este del dominio en coordenadas proyectadas (por defecto 679476 si domain es True)"
                required: false
              - in: query
                name: south
                schema:
                  type: number
                description: "Límite sur del dominio en coordenadas proyectadas (por defecto 4609612 si domain es True)"
                required: false
              - in: query
                name: north
                schema:
                  type: number
                description: "Límite norte del dominio en coordenadas proyectadas (por defecto 4617648 si domain es True)"
                required: false
            summary:
              "Descarga de fichero shapefile con nox a partir del viario en la base de datos"
            responses:
              200:
                description: "Success"
              400:
                description: "Bad request"
              500:
                description: "Internal server error"
            operationId: "src.app.roads.shp"
            tags:
                - "roads"
```

Listing 3.1: Extracto del fichero YAML para la descripción de la llamada API mostrada en la figura 5.2.

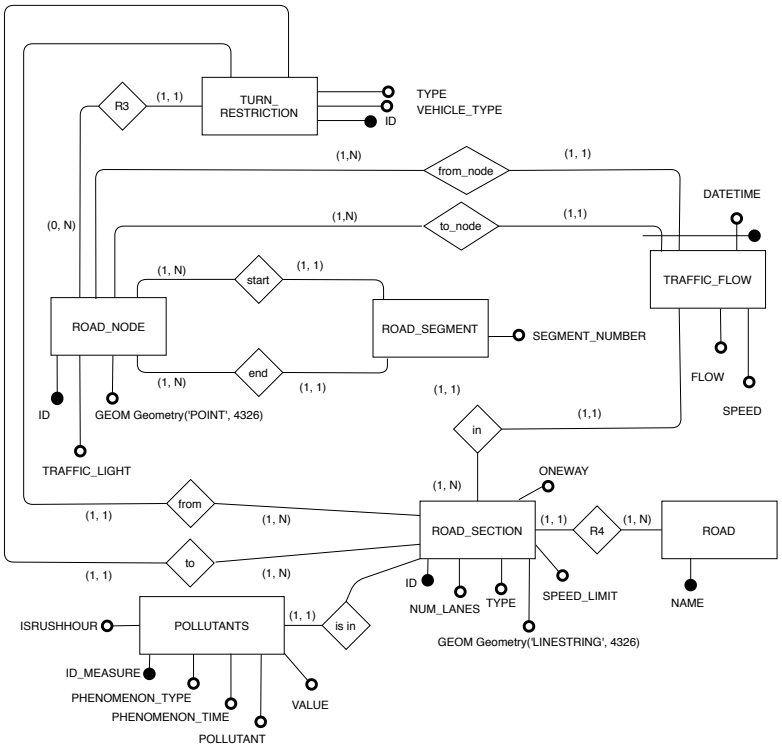


Figura 3.3: Diagrama Entidad-Relación de las tablas de red viaria de la base de datos

3.2. Red viaria (*roads*)

Tal y como hemos visto en el capítulo 2, las emisiones por segmento se guardan en la base de datos a partir del modelo de emisiones y se asocian a los segmentos de calle guardados en la base de datos. Estos segmentos fueron extraídos de OpenStreetMap[28], insertándose en la base de datos en la fase de entrada del proyecto mediante un script Python que realizaba una consulta a Overpass API[29], que es una API de solo lectura optimizada para minería de datos de OpenStreetMap, permitiendo realizar consultas en un lenguaje de consultas, descargando así únicamente los datos necesarios para el proyecto; y posteriormente transformándolos al formato guardado en la base de datos.

En la figura 3.3 se puede observar el modelo Entidad-Relación de la parte necesaria de la base de datos del proyecto. Está montada con el software sistema gestor de bases de datos PostgreSQL[30] junto con el complemento PostGIS[31], que la convierte en una base de datos espacial, añadiendo nuevos tipos de datos y funciones e índices pensados para trabajar eficientemente

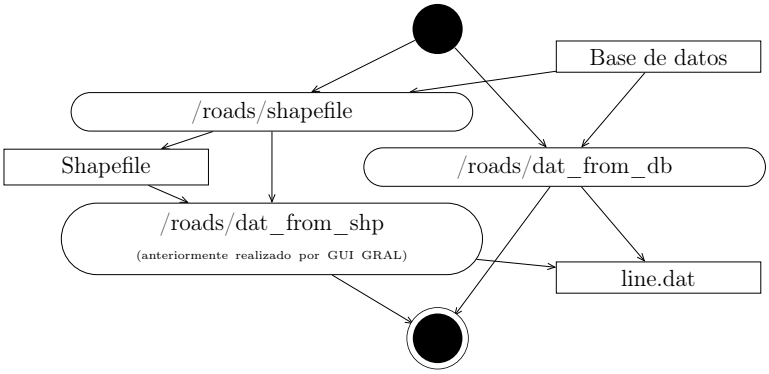


Figura 3.4: Flujo de las llamadas de red viaria de la aplicación de publicación interna de datos

con información espacial.

Para consultar y descargar la información de la base de datos aprovechando las funciones de PostGIS, además de para poder trabajar mejor con datos espaciales en Python, se ha utilizado la librería GDAL[32][33].

Las funciones de Red Viaria (*roads*) se centran principalmente en la transformación y descarga de datos de los niveles de NOx simulados para cada segmento de calle del viario urbano.

La primera función, *Descarga de fichero shapefile con nox a partir del viario en la base de datos (roads/shapefile)*, une las tablas de segmentos de red viaria y de emisiones por segmento y las transforma a formato ESRI Shapefile para su visualización en programas SIG. Se escogió este formato al tener la interfaz gráfica de GRAL ya implementadas funciones que transforman Shapefile a un fichero line.dat que sirve como entrada para las simulaciones. Posteriormente se implementaron las otras dos funciones, que simulan este paso hecho por la interfaz gráfica de GRAL para poder automatizar más el proceso: primero la descarga del fichero line.dat a partir del Shapefile (*roads/dat_from_ shp*) generado por la anterior función, y posteriormente la descarga del fichero line.dat a partir del viario en la base de datos (*roads/-dat_from_ db*), que ahorra este paso intermedio por el fichero Shapefile.

Todas las funciones contemplan los parámetros expuestos en la tabla 3.1, que nos permiten definir mejor la salida que queremos. Primero tenemos el parámetro *type_ except*, con el que podemos elegir eliminar ciertos tipos de vías, siguiendo la clasificación[34] de OpenStreetMap. Esto posibilita variar el nivel de detalle de la simulación si se comprueba que tarda demasiado en ejecutar, quitando vías de menor importancia.

Los demás parámetros se refieren a limitar el dominio de la salida a un área determinada, con un rectángulo (denominado *Bounding Box* en inglés)

delimitado por los valores de los cuatro puntos cardinales que se deseen (parámetros *west*, *east*, *south* y *north*), el parámetro *domain* sirve para activar o desactivar esta función. Por defecto se limita con un rectángulo que incluye la mayor parte del casco urbano de Zaragoza.

Esta limitación se ha realizado con la función específica de PostGIS *ST_MakeEnvelope* en el caso de la función que genera el Shapefile o con la clase BoundingBox para las funciones que generan el fichero *line.dat*. Se puede ver el ejemplo de la función *generate_dat*, que se llama desde *dat_from_db* y *dat_from_shp*, en el código 3.2.

Cada vía de la base de datos se corta en segmentos entre nodos consecutivos y se verifica que ambos nodos del segmento estén dentro del rectángulo. Antes de ello, se convierten las coordenadas de cada vía a proyectadas EPSG:25830 - ETRS89 / UTM zone 30N[35], que es el sistema que GRAL utiliza, desde EPSG:4326 / WGS84[36], que OpenStreetMap usa y en el que se han guardado los valores en la base de datos.

```
def generate_dat(layer: ogr.Layer, type_except="", domain=BoundingBox(-np.inf, np.inf, -np.inf, np.inf)):
    valid_except = _except(type_except, "arr")

    source = osr.SpatialReference()
    source.ImportFromEPSG(4326)
    target = osr.SpatialReference()
    target.ImportFromEPSG(25830)

    transform = osr.CoordinateTransformation(source, target)

    file_path = "line.dat"
    with open(file_path, "w") as file:

        file.write("Generated: \nGenerated: \nGenerated: \nGenerated: \nStrName,Section,Sourcegroup,x1,y1,z1,x2,y2,"
                   "z2,width,noiseabatementwall,Length[km],--,pollutant[kg/(km*h)],--,--,--,--,--,deposition data\n")
        feature = layer.GetNextFeature()
        while feature:
            if feature.GetField("type") not in valid_except:
                _id = feature.GetFID()
                geom: ogr.Geometry = feature.GetGeometryRef()
                geom.Transform(transform)
                points = geom.GetPoints()
                width = feature.GetField("width")
                num_lanes = feature.GetField("num_lanes")
                try:
                    nox = str(feature.GetField("nox"))
                except Exception as e:
                    print("Nox not found:", e)
                    nox = str(round(num_lanes * 0.1, 1)) if num_lanes is not None else "0.1"
                for i in range(len(points) - 1):
                    x1 = points[i][0]
                    y1 = points[i][1]
                    x2 = points[i + 1][0]
                    y2 = points[i + 1][1]
                    if domain.is_inside(x1, y1) & domain.is_inside(x2, y2):
                        # Segments are added if they are completely inside the domain BoundingBox
                        file.write(",".join(["Line " + str(_id), "1", "1", str(round(x1, 1)), str(round(y1, 1)), "0", str(round(x2, 1)), str(round(y2, 1)), "0", str(width), "-3", "0", "0", nox, "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0"]) + "\n")
            feature = layer.GetNextFeature()

    return flask.send_from_directory(os.path.dirname(__file__), file_path, as_attachment=True)
```

Listing 3.2: Función generate_dat del fichero src/app/roads.py

Nombre de parám.	Tipo de parám.	Descripción
type_except	string	Tipos de vía a eliminar, separados por ","(sin espacios). Ejemplo: track,service
domain	boolean	True si se desea usar un rectángulo para limitar el dominio de la salida, False en caso contrario (por defecto True)
west	float	Límite oeste del dominio en coordenadas proyectadas EPSG:25830 - ETRS89 / UTM zone 30N (por defecto 667840 si domain es True, si domain es False el parámetro no se tiene en cuenta)
east	float	Límite este del dominio en coordenadas proyectadas EPSG:25830 - ETRS89 / UTM zone 30N (por defecto 682136 si domain es True, si domain es False el parámetro no se tiene en cuenta)
south	float	Límite sur del dominio en coordenadas proyectadas EPSG:25830 - ETRS89 / UTM zone 30N (por defecto 4607928 si domain es True, si domain es False el parámetro no se tiene en cuenta)
north	float	Límite norte del dominio en coordenadas proyectadas EPSG:25830 - ETRS89 / UTM zone 30N (por defecto 4618140 si domain es True, si domain es False el parámetro no se tiene en cuenta)

Tabla 3.1: Parámetros de las funciones de extracción de red viaria

El fichero generado line.dat tiene un formato tipo csv, con columnas indicadas en la tabla 3.2.

Columna	Descripción
StrName	Idenficador de la vía. Rellenado con "Line" + id interno
Section	Sección de la vía
Sourcegroup	Grupo de emisiones
x1, y1, z1, x2, y2, z2	X, Y, Z min. y máx. del rectángulo envolvente (*bounding box*) del segmento de vía
width	Anchura de la vía
noiseabatementwall	Altura de la barrera de abatimiento del sonido (se ha usado una media de la altura de los edificios)
Length[km]	–
–	–
pollutant[kg/(km*h)]	Emisiones del contaminante NOx estimadas en las fases intermedias del proyecto
–	–
–	–
–	–
–	–
–	–
deposition data	–

Tabla 3.2: Columnas del archivo generado line.dat

3.3. Datos meteorológicos (*weather*)

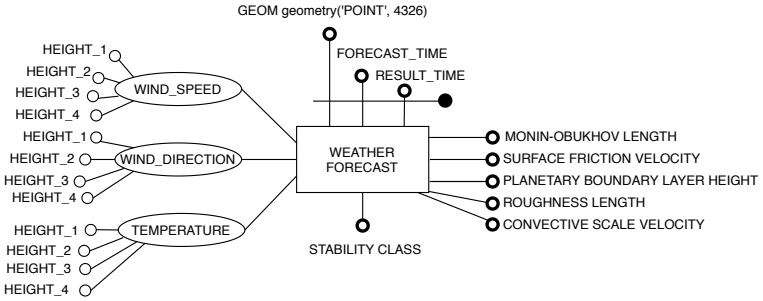


Figura 3.5: Diagrama Entidad-Relación de las tablas de predicción meteorológica de la base de datos

Dentro de la base de datos también se guardan datos de predicciones meteorológicas actualizados diariamente mediante un script Python que realiza peticiones al servidor THREDDS de MeteoGalicia[37]. Ahí, pide los datos del modelo WRF para las próximas 96 horas en el área de Zaragoza, realiza las transformaciones necesarias y los inserta en la tabla `WEATHER_FORECAST` que vemos en el esquema Entidad-Relación de la figura 3.5.

Para su descarga existe una única función que genera dos archivos a partir de la información guardada en la base de datos del proyecto. Estos son meteopgt.all y mettimeseries.dat, y se utilizan posteriormente como entrada de la simulación en GRAL.

La función incluye los parámetros datetime (fecha y hora de inicio de predicción) y hours (número de horas para las que se realizará la predicción), y consiste en realizar una consulta a la base de datos para obtener la información sobre la última predicción disponible para esa hora y hacer pequeñas transformaciones a los datos para rellenar ambos ficheros.

Se pueden observar los valores de las columnas de los ficheros (que tienen un formato similar a csv) en las tablas 3.3 y 3.4. Ambos tienen una línea por cada hora a partir de datetime, hasta llegar a hours líneas. A esto hay que añadir una línea extra en meteopgt.all al inicio, donde se hace referencia a la altura del anemómetro (height_1 en la base de datos) e información sobre la categorización de nuestros datos, que rellenamos con 0 al no estar categorizados los que usamos.

Columna	Descripción
Wind direction sector	Dirección del viento en la altura del anemómetro / 10
Wind speed class	Velocidad del viento en la altura del anemómetro
stability class	Clase de estabilidad
frequency	Frecuencia con la que se dan esas condiciones específicas de meteorología en el histórico de datos meteorológicos

Tabla 3.3: Columnas del archivo generado meteopgt.all

Columna	Descripción
Date	Mes y día
Time	Hora
Wind speed class	Velocidad del viento en la altura del anemómetro
Wind direction sector	Dirección del viento en la altura del anemómetro / 10
stability class	Clase de estabilidad

Tabla 3.4: Columnas del archivo generado mettimeseries.dat

3.4. Factores de emisión (*emissions*)

El Ayuntamiento de Zaragoza cuenta con sensores permanentes de medición de tráfico, que permiten conocer la intensidad del tránsito en diversos puntos de la ciudad y para cada hora. Estas mediciones se subieron a la base de datos (ver figura 3.6) mediante un script Python que leía y transformaba los valores en ficheros Excel con las mediciones en los últimos años que el Servicio de Movilidad del Ayuntamiento nos proporcionó, además de anteriormente haber subido las ubicaciones de las estaciones asociándolas al viario ya presente en la base de datos.

Con esta llamada */emissions* se desea calcular los factores de emisión para una franja temporal, esto es, una estimación de a qué horas se cree que se van a generar las mayores emisiones para las simulaciones que genera GRAL, basado siempre en datos históricos de años pasados al no estar disponible la

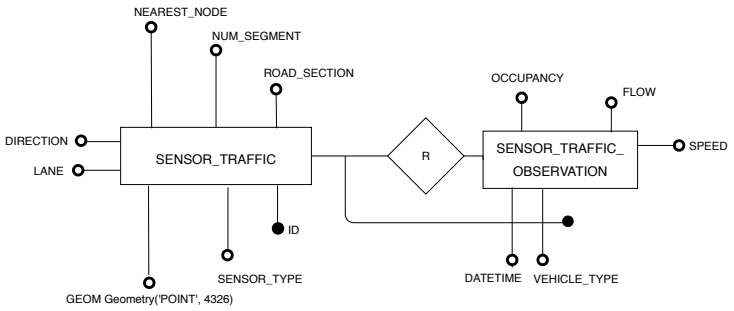


Figura 3.6: Diagrama Entidad-Relación de las tablas de sensores permanentes de tráfico de la base de datos, que sirven para calcular los factores de emisión

información de los sensores en tiempo real. Incluye los parámetros date_from (fecha de inicio para el fichero que queremos generar) y date_to (fecha de final).

Las estimaciones se realizan tal que para cada día sobre el que queremos calcular las emisiones generadas por el tráfico buscamos un día equivalente en el año anterior, restando 364 días para conseguir mismo día de la semana. Además, también tenemos en cuenta si el día para el que se realizan las estimaciones es festivo mediante la librería holidays y su función para chequear festivos por ubicación (ej: holidays.CountryHoliday("ES", "ARG") para Aragón) y seguimos la tabla 3.5.

		Fecha a calcular es festivo	
		Sí	No
Fecha similar es festivo	Sí	Fecha similar es correcta	Echar una semana atrás la fecha similar y repetir procedimiento
	No	Buscar el festivo con el mismo nombre el año anterior, si no lo encuentra suponemos fecha similar correcta	Fecha similar es correcta

Tabla 3.5: Decisiones tomadas sobre fechas para encontrar una fecha similar en el año anterior

Una vez tenemos el día similar, se genera la información sobre intensidad de tráfico para cada hora de ese día en términos porcentuales según las mediciones recopiladas en la base de datos, siendo 1 (100 %) la hora con mayor flujo de vehículos, y calculándose la fracción correspondiente para las demás horas. Podemos ver un fichero de ejemplo en el código 3.3.

```
Day.Month     Hour     1
26.07         0        0.18620584265084175
26.07         1        0.10363446501795584
26.07         2        0.06759544609428725
26.07         3        0.05351093905203372
26.07         4        0.07189975294806815
26.07         5        0.1959605735679902
26.07         6        0.40643863179074446
26.07         7        0.7905661819015358
26.07         8        0.8976389985482516
26.07         9        0.8400529760843543
26.07         10       0.7951251814685582
26.07         11       0.8388304510607951
26.07         12       0.8688332526806408
26.07         13       0.9502075745612918
26.07         14       1.0
26.07         15       0.9500802282046711
26.07         16       0.8042177113312788
26.07         17       0.8419377021623411
26.07         18       0.9172248681965209
26.07         19       0.8975116521916308
26.07         20       0.8126225708682474
26.07         21       0.6832896110842269
26.07         22       0.532995441000433
26.07         23       0.338461146626595
```

Listing 3.3: Archivo emissions_timeseries.txt de ejemplo para las 24 horas de un mismo día

4 | Publicación de datos abiertos

En la figura 4.1 podemos ver los componentes que influyen en la publicación de datos abiertos del proyecto. Se puede interpretar también como un diagrama de flujo de abajo a arriba, siendo GeoServer el componente que se conecta directamente con la base de datos del proyecto y sirve los datos mediante servicios de visualización y descarga compatibles con las especificaciones de OGC, y subiendo de forma automatizada a los servicios de ámbito superior para alcanzar cada vez a una mayor parte de la ciudadanía. El paso intermedio por la pasarela CKAN-Zaguan y Zaguan se realiza solo en Zaragoza, en los demás casos el servidor CKAN de nivel superior recolecta directamente del local, se explican los casos concretos en la sección 5.2.

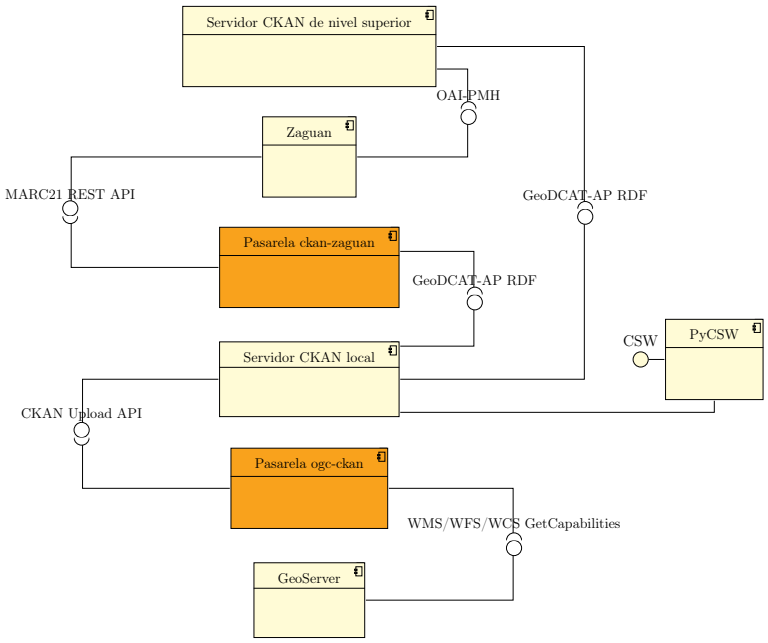


Figura 4.1: Diagrama de componentes de publicación de datos abiertos

En la sección 4.1 se describe de forma resumida el perfil de metadatos que se ha utilizado.

En la sección 4.2 se describe el flujo de trabajo para publicar en un portal de tipo CKAN la información recolectada a través de los servicios OGC.

En la sección 4.3 se describe el flujo de trabajo para publicar en el repositorio institucional de documentos Zaguan. Este paso sólo es realizado en el caso de Zaragoza.

Por último, en la sección 4.4 se describen los pasos realizados para publicar en un servicio CSW.

4.1. Perfil de metadatos basado en GeoDCAT-AP

ISO 19115 es el estándar internacional para metadatos geográficos propuesto por la Organización Internacional de Normalización (ISO por sus siglas en inglés)[38], que ha sido ampliamente adoptado durante la última década en la comunidad de de la información geográfica en sectores tanto públicos como privados.

Sin embargo, en el dominio de datos abiertos se necesitan esquemas de metadatos más generales y simples para facilitar la publicación de conjuntos de datos de diferentes disciplinas en el mismo repositorio de metadatos. DCAT es el acrónimo del vocabulario del Catálogo de datos del W3C[21] y puede ser considerado como un núcleo básico y general de propiedades de metadatos compartidas por los diferentes esquemas de metadatos usados en diversas iniciativas de Datos Abiertos. En el caso europeo, la Unión Europea propuso en 2013 DCAT-AP [20], una especificación basada en DCAT para describir conjuntos de datos del sector público en Europa. Comparado con DCAT, DCAT-AP provee definiciones más estrictas de catálogos, conjuntos de datos, distribuciones y otros objetos.

Como se ha mencionado en la introducción, en el contexto de este proyecto hemos seleccionado GeoDCAT-AP v1.01 [19]. Esta extensión de DCAT-AP está diseñada para la descripción de datos espaciales y sus propiedades de metadatos tienen un mapeo exacto con los elementos principales de los metadatos ISO 19115. Este mapeo garantiza la transformación de registros GeoDCAT-AP a su equivalente en ISO 19115 conforme con los requerimientos INSPIRE[22][39].

La descripción de conjuntos de datos según GeoDCAT-AP está principalmente pensada en proveer información sobre tres entidades principales: un *Catálogo* (Catalog) que se publica mediante un portal de Datos Abiertos con-

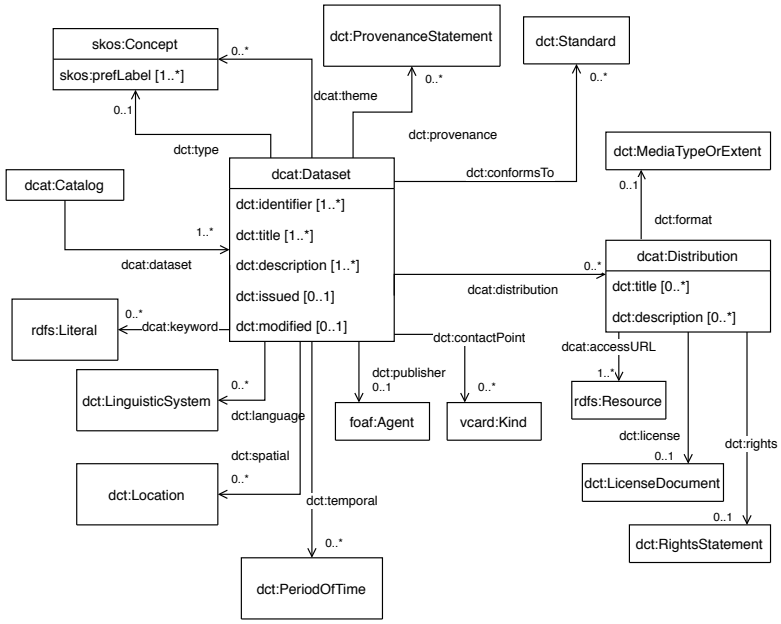


Figura 4.2: Entidades y propiedades usadas de GeoDCAT-AP

teniendo *Conjuntos de datos* (Datasets) y *Distribuciones* (Distributions), las formas asociadas para cada conjunto de datos. Además, GeoDCAT-AP hace una distinción entre propiedades principales (core) y extendidas (extended). El conjunto principal es la selección de propiedades de metadatos DCAT-AP que tienen una conexión directa con los metadatos ISO 19115 e INSPIRE. En algunos casos, estas propiedades adicionales pertenecen a otros vocabularios de metadatos. En otros casos, aunque las propiedades pertenezcan a DCAT-AP, son clasificadas como extendidas al proveer tan solo una conexión o mapeo parcial con ISO 19115 e INSPIRE.

La figura 4.2 muestra un diagrama UML con las propiedades de GeoDCAT-AP que son necesarias para describir conjuntos de datos y distribuciones en TRAFAIR. La mayoría de estas propiedades pertenecen al conjunto principal de GeoDCAT-AP. Las únicas excepciones son las propiedades *dct:type* de *Conjuntos de datos* (Datasets) y la propiedad *dct:description* de *Distribuciones* (Distributions). *dct:type* es empleado para indicar si el recurso descrito es un conjunto de datos o una serie de conjuntos de datos. *dct:description* permite la descripción de la resolución espacial de las distribuciones asociadas.[1]

[1]A pesar de que GeoDCAT-AP propone *rdfs:comment* como una propiedad provisional para rellenar esta información de resolución, no hay un mapeo directo de esta propiedad a campos de CKAN y por lo tanto hemos considerado *dct:description* como una alternativa

4.2. Servicio de publicación automática de metadatos en servidores CKAN a partir de servicios OGC

La figura 4.3 muestra un diagrama de actividad con los cinco pasos principales del flujo de trabajo que hemos propuesto para la publicación de Datos Abiertos Espaciales. Para los pasos 1, 3 y 4 hemos desarrollado software para automatizar lo máximo posible la generación y publicación de metadatos. Los pasos 2 y 5 son conseguidos gracias al uso de programas ya existentes.

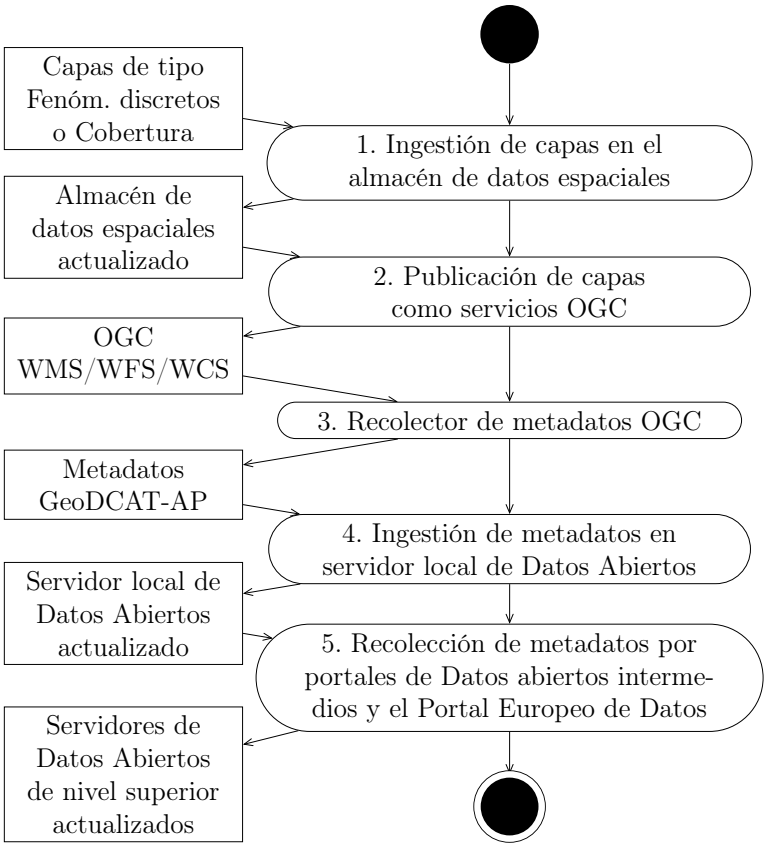


Figura 4.3: Flujo de trabajo para la publicación de Datos Abiertos Espaciales

El primer paso es la ingestión de capas en un almacén de datos espaciales. Geoserver[8] es el software elegido para gestionar la publicación de capas de datos espaciales, tanto para fenómenos discretos (en inglés: Feature Type) como para datos de cobertura (Coverage), ya que permite una fácil conexión

válida.

con la base de datos, permitiendo seleccionar las tablas que se necesiten o una consulta como capas de GeoServer. También posibilita la creación de metadatos y estilos personalizados para cada capa para cumplir las necesidades del proyecto y facilita la introducción de atributos comunes como la información de contacto del administrador del servidor.

En el contexto del proyecto TRAFAIR se ha desarrollado software específico en los lenguajes Java y R[2] para la ingestión de capas de datos discretos (soportadas en una base de datos espacial) y coberturas en GeoServer a través de su API REST[40][41]. Con respecto a la generación de metadatos, este software se encarga de enviar a la API REST los valores apropiados para las etiquetas enumeradas en la columna *Geoserver* de la tabla 4.1.

El segundo paso es la publicación de capas como servicios OGC. Este paso es conseguido directamente gracias al software de GeoServer, que provee acceso a capas a través de diferentes servicios OGC. Capas de fenómenos discretos como observaciones del tráfico y de sensores de calidad del aire pueden ser descargadas desde un servicio WFS. En el caso de coberturas para monitorización de la calidad del aire (interpolaciones de observaciones de sensores) o coberturas para la predicción de calidad (el resultado de un modelo GRAL), un servicio WCS es brindado. Más allá de WFS y WCS, algunas capas también están disponibles para servir representaciones de mapas generadas del lado del servidor utilizando un servicio WMS.

El tercer paso es la recolección de metadatos de los servicios OGC a través de su operación *GetCapabilities*. Para implementar este paso, hemos desarrollado un programa en Python que aprovecha la librería OWSLib[42], un paquete de cliente para interactuar con las interfaces de los servicios OGC y sus modelos de contenido relacionados. Este software interacciona con la interfaz OGC, en vez de la API REST de GeoServer, ya que queríamos hacer este software lo suficientemente escalable para integrar en un futuro otras capas gestionadas por otros paquetes software distintos de GeoServer. OWSLib transforma los modelos de los servicios a clases de Python, pudiendo directamente trabajar de forma cómoda con objetos de dichas clases y recopilar los atributos necesarios para nuestra pasarela.

[2]Este software no ha sido desarrollado por el autor del trabajo, sino por otros compañeros del proyecto y por lo tanto no entra dentro de este TFG.

Geoserver	**OWSLib**	**CKAN**	**GeoDCAT-AP**
featureType/name, coverage/name	*layerName*	extra:identifier	Dataset/dct:identifier
featureType/title, coverage/title	contents[*layerName*].title	title	Dataset/dct:title
featureType/description, coverage/description *(software en el paso 1 introduce descripciones predeterminadas de acuerdo a patrones de nombres)*	contents[*layerName*].abstract	notes	Dataset/dct:description
	("series" para servicios OGC con dimensión temporal o "dataset" sin dimensión temporal)	extra:dcat_type	Dataset/dct:type
	(idioma por defecto propuesto en el paso 3)	extra:language	Dataset/dct:language
	(temas de datos por defecto de INSPIRE y categorías de temas ISO 19115 propuestos en el paso 3)	extra:theme	Dataset/dcat:theme
(algunas palabras clave son introducidas automáticamente por Geoserver)	contents[*layerName*].keywords	tags	Dataset/dcat:keyword
(computado automáticamente por Geoserver)	contents[*layerName*].boundingBoxWGS84	extra:spatial	Dataset/dct:spatial
(fecha de inicio y fecha final son actualizadas automáticamente por Geoserver)	contents[*layerName*].timepositions	extra:temporal_start + extra:temporal_end	Dataset/dct:temporal
		extra:issued *(insertado automáticamente con la primera ingestión en CKAN)*	Dataset/dct:issued
		extra:modified *(actualizado automáticamente con cada actualización de conjunto de datos en CKAN)*	Dataset/dct:modified
	(provenencia por defecto propuesta en el paso 3)	extra:provenance	Dataset/dct:provenance
	(conformidad INSPIRE por defecto y sistema de referencia de coordenadas propuesto en el paso 3)	extra:conforms_to	Dataset/dct:conformsTo
(la información de contacto es introducida directamente por los administradores en la página de configuración de GeoServer)	contents[*layerName*].provider.contact. organization	extra:publisher_name	Dataset/dct:publisher
	contents[*layerName*].provider.contact.name + contents[*layerName*].provider.contact.email	extra:contact_name + extra:contact_email	Dataset/dcat:contactPoint
(URL de servicio OGC generada automáticamente por Geoserver)	*(OGC service URL)*	resource:url	Distribution/dcat:accessURL
featureType/name, coverage/name	*layerName*	resource:name	Distribution/dct:title
featureType/serviceConfiguration, coverage/serviceConfiguration	*("wfs", "wcs" o "wms" según tipo de servicio OGC)*	resource:format	Distribution/dct:format
	(licencia por defecto propuesta en paso 3)	resource:license	Distribution/dct:license
	(derechos por defecto propuestos en paso 3)	resource:rights	Distribution/dct:rights
	(resolución por defecto propuesta para los conjuntos de datos del proyecto)	resource:description	Distribution/dcat:description

Tabla 4.1: Correspondencia entre etiquetas de GeoServer (contenidas en el cuerpo de una petición POST para crear una capa de fenómenos discretos o cobertura), campos de la capa recuperado con OWSLib de una respuesta GetCapabilities, etiquetas de CKAN (contenidas en el cuerpo de una petición POST para crear un conjunto de datos), y propiedades GeoDCAT-AP.

En la figura 4.4 se pueden observar las clases de este programa Python, que hemos denominado Pasarela OGC-CKAN.

El algoritmo detrás de este software genera una instancia de *Dataset* (conjunto de datos) por cada capa publicada en los servicios WFS o WCS. Además, cada *Dataset* tiene al menos una instancia de *Distribution* (distribución) asociada en la forma de un enlace a un servicio WFS o WCS.

En algunos casos, si la capa es también renderizada a través de un WMS, una segunda distribución enlazando al servicio WMS es generada. La columna *OWSLib* en la tabla 4.1 muestra los campos recuperados por el paquete OWSLib para generar la propiedad GeoDCAT-AP correspondiente.

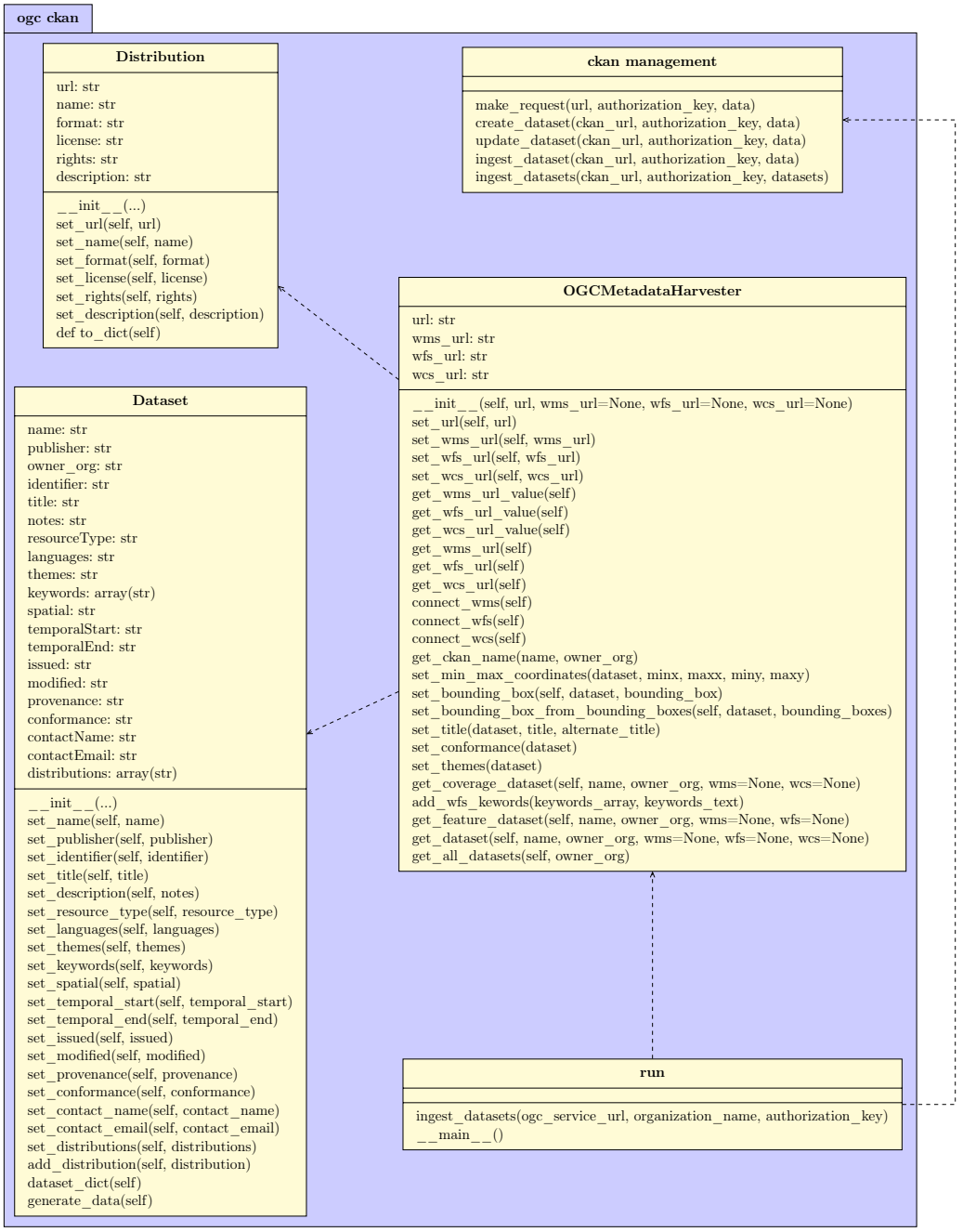


Figura 4.4: Diagrama de clases pasarela OGC-CKAN

El cuarto paso es la ingestión de metadatos en el servidor de Datos abiertos de la institución en cargo de la publicación de los datos de calidad del aire en el área local. Como continuación del software en el paso anterior, nuestro

programa en Python transforma la información recopilada en el paso anterior a un diccionario con los objetos requeridos para construir un conjunto de datos con sus recursos asociados, que son inmediatamente insertados en un servidor local de Datos Abiertos basado en CKAN a través de su API REST[43]. La columna *CKAN* en la tabla 4.1 indica las etiquetas que son usadas en esta estructura diccionario de datos para generar después metadatos RDF basados en GeoDCAT-AP (se ha seguido el mapeo explicado en [23], sección *RDF DCAT to CKAN dataset mapping*).

El paso final es la recolección de metadatos en los servidores locales desde los portales de Datos Abiertos regionales y nacionales hasta que al final los metadatos son últimamente recolectados por el Portal Europeo de Datos. Este paso está más allá del alcance del proyecto TRAFAIR. Sin embargo, suponemos que los portales de nivel superior se basan en la tecnología CKAN (o tienen un mecanismo similar para la recolección de catálogos suscritos de nivel inferior).

Por una parte, el plugin *ckanext-dcat* de CKAN permite la publicación de metadatos de conjuntos de datos como RDF de acuerdo con los vocabularios DCAT-AP. Y por la otra parte, el complemento *ckanext-harvest* de CKAN permite recoletar los contenidos de diferentes tipos de fuentes de catálogos.

4.3. Publicación de metadatos en Zaguan

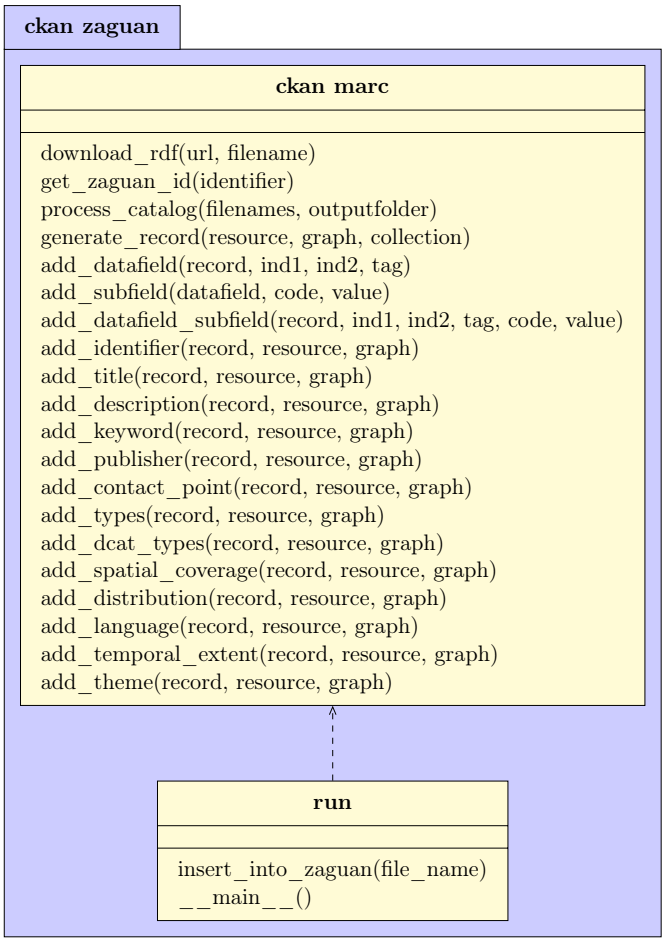


Figura 4.5: Diagrama de clases pasarela CKAN-Zaguan

Con respecto a los repositorios de datos abiertos, actualmente la Universidad de Zaragoza cuenta con el repositorio institucional Zaguan[44], montado sobre el software Invenio[45]. Este repositorio está pensado principalmente para elementos bibliográficos y usa el estandar MARC21[46].

Los conjuntos de datos de Zaguan son después regularmente volcados en Aragón Open Data[47], de ahí al portal de datos abiertos del Gobierno de España[48] y por último al Portal Europeo de Datos[18]. Los tres portales usan principalmente el software CKAN[12].

En la figura 4.5 se puede ver las clases que componen esta pasarela. La función principal (*main*) descarga un catálogo RDF desde un punto de acceso (con la función *download_rdf*), en este caso desde nuestro servidor CKAN

local. Posteriormente llama a *insert_into_zaguan*, que procesa los conjuntos de datos y series del catálogo descargado usando las funciones de la clase *ckan_marc*, que genera sus metadatos correspondientes en MARC21, y sube los datos generados a Zaguan mediante su API de ingestión.

En la tabla 4.2 encontramos la correlación que se ha decidido implementar entre los metadatos en GeoDCAT-AP y en MARC21. Se ha intentado en todo momento emparejar cada campo con uno similar en el otro estándar, y que luego tuviese salida en la pasarela que sube los datos de Zaguan a Aragón Open Data (sobre la que desde el proyecto no tenemos control), no siendo posible para todos los metadatos presentes en el servidor CKAN local.

Por último, se manda un email automáticamente al administrador de Zaguan para advertir sobre la subida para que puedan tener un registro de las cargas que se hacen desde el proyecto.

GeoDCAT-AP	**MARC21**
Dataset/dct:identifier	037 ##$a, 970 ##$a (ids `'TRAFAIR-2020-XXX'`, mapeando cada XXX al identificador GeoDCAT-AP con números ascendentes)
Dataset/dct:title	245 ##$a
Dataset/dct:description	530 3#$a
Dataset/dct:type	655 #7$a (incluyendo en 655 #7$2 el valor `'inspire'`)
Dataset/dct:language	041 ##$a
Dataset/dcat:theme	654 ##$a (incluyendo en 654 ##$2 el valor `'inspire'`)
Dataset/dcat:keyword	653 1#$a
Dataset/dct:spatial	-
Dataset/dct:temporal	518 ##$a
Dataset/dct:issued	260 ##$c
Dataset/dct:modified	-
Dataset/dct:provenance	-
Dataset/dct:conformsTo	-
Dataset/dct:publisher/foaf:name	260 ##$b
Dataset/dcat:contactPoint/vcard:fn	700 ##$a
Distribution/dcat:accessURL	856 4#$u
Distribution/dct:title	856 4#$n
Distribution/dct:format	856 4#$y
Distribution/dct:license	540 ##$u, además para licencias Creative Commons: 540 ##$2=`'cc'`, 540 ##$a=(modificador), 540 ##$b=`'Creative Commons'`, 540 ##$c=(versión), 540 ##$f=`'CC'` + (modificador) + (versión), 540 ##$9=`'info:eu-repo/semantics/openAccess'`
Distribution/dct:rights	506 X#$u, siendo X 1 si existen limitaciones al acceso o 0 si no, y si no además: 506 0#$a=`'Access copy available to the general public'`, 506 0#$f=`'Unrestricted'`
Distribution/dcat:description	-
-	980##$X (categorías y subcategorías del portal Zaguan, siendo X ascendente alfabéticamente empezando por la a)

Tabla 4.2: Correspondencia entre etiquetas de GeoDCAT-AP y MARC21 para la carga en Zaguan

4.4. Publicación de metadatos en un catálogo CSW

Para poner en marcha un servicio de catálogo que permitiese acceder a los metadatos según el estándar ISO 19115[38] y que este servicio fuese compatible con la especificación OGC Catalog Services for the Web (CSW)[5] se contemplaron diversas opciones. Primero se consideró la posibilidad de instalar un complemento para catálogos CSW en Geoserver, versión que se estuvo usando en el proyecto de forma provisional durante un tiempo pero que al final fue descartada al no llegar al nivel de metadatos que requeríamos: los metadatos ISO 19115 autogenerados por GeoServer apenas informaban del título de las capas.

También se planteó el uso de GeoNetwork[49], que es un software ampliamente utilizado para la puesta en marcha de servidores y clientes de búsqueda de catálogos de metadatos geográficos. Sin embargo, tenía el inconveniente de no facilitar la sincronización directa con los metadatos que se estaban generando de forma automática según el proceso descrito en la sección 4.3 Geonetwork permite extraer algo de información de las capas gestionadas por Geoserver, pero el resto de elementos de metadatos se debe editar manualmente.

Finalmente, nos acabamos decantando por pyCSW[13], que proporciona un servidor de catálogo compatible con la especificación CSW desarrollado en Python, que se puede poner en marcha y gestionar de forma rápida y sencilla. Esta decisión fue reforzada al contar con una integración con CKAN a través de ckanext-spatial[14]. Los metadatos GeoDCAT-AP generados con el proceso descrito en la sección 4.3 se podían reconvertir de forma automática a metadatos ISO 19115.

PyCSW cuenta con una base de datos separada a la de CKAN, a la que se hacen cargas periódicas ejecutando el comando *ckan-pycsw* de ckanext-spatial.

En el anexo A se incluye la instalación y el despliegue de pycsw en detalle, además del comando para la carga.

5 | Puesta en marcha y validación

5.1. Publicación interna de datos

En la figura 5.1 podemos ver la interfaz gráfica OpenAPI-Swagger de la aplicación de publicación interna de datos con los detalles de cada una de las funciones de la API a las que llama.

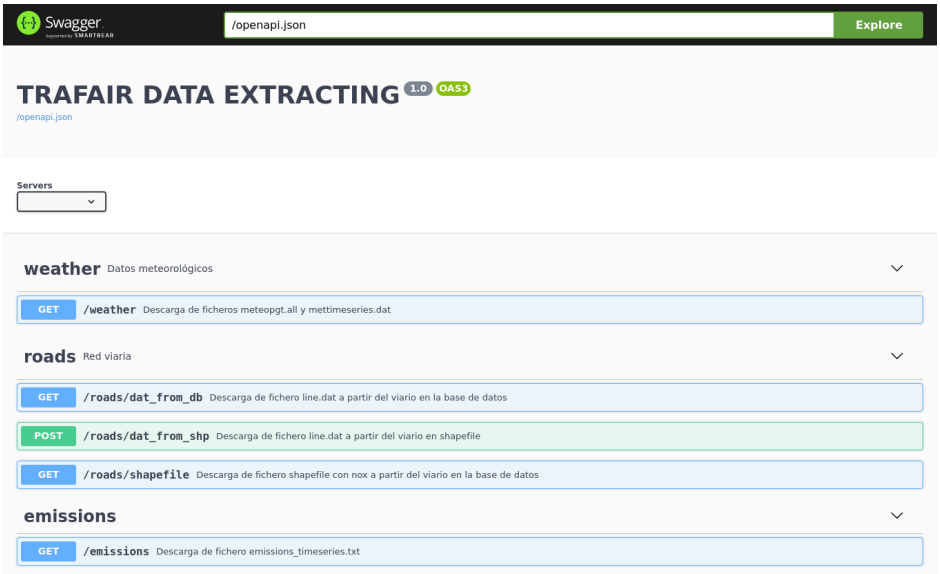


Figura 5.1: Interfaz gráfica OpenAPI de la aplicación de publicación interna de datos

En la figura 5.2 se observa el detalle de una de las funciones, donde vienen reflejados los parámetros con su nombre, descripción, tipo y su obligatoriedad (o no) de uso. También se muestran los diferentes códigos de respuesta que se pueden recibir.

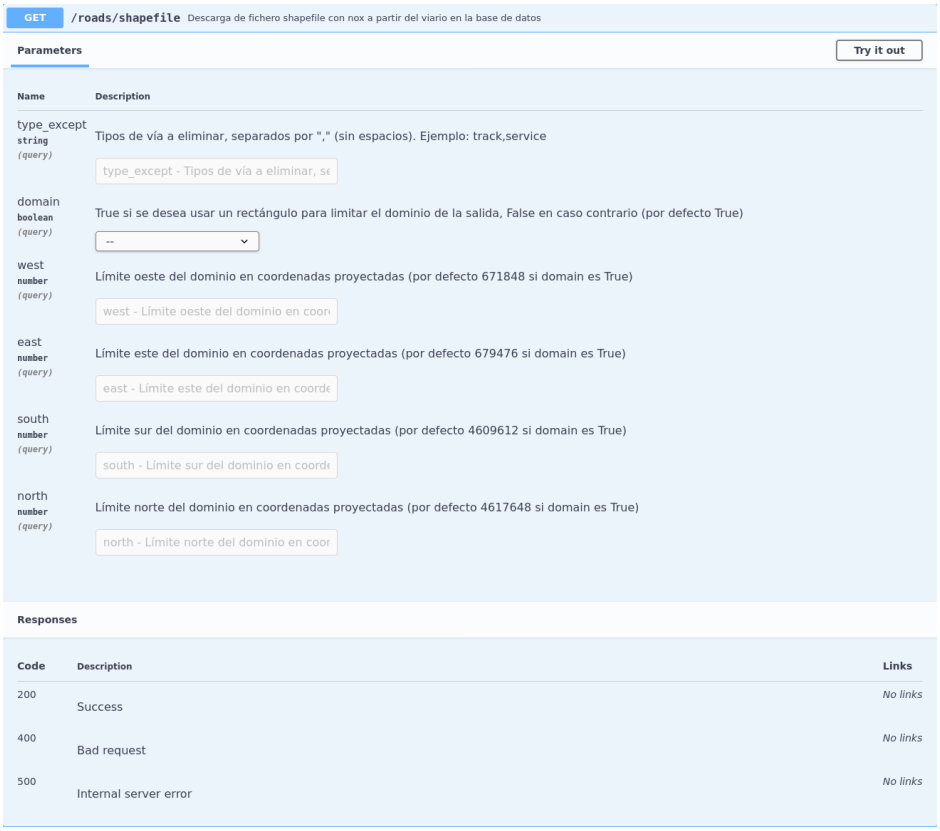


Figura 5.2: Detalle de función con parámetros dentro de la interfaz OpenAPI-Swagger

Además de poder utilizar la interfaz de usuario presentada, también se hace constar que esta aplicación se puede invocar mediante peticiones cURL a la API de publicación interna de datos. Gracias a este tipo de peticiones, se puede automatizar la inserción de los ficheros generados como entrada en el simulador GRAL cuando se programa la ejecución diaria de los modelos de predicción.

En la base de datos encontramos 8345 segmentos de vía en el área de Zaragoza limitada por el rectángulo por defecto, además de 31725 predicciones meteorológicas y 771480 observaciones de flujo de tráfico de 46 estaciones fijas.

En la figura 5.3 se puede ver la visualización con la herramienta QGIS de un ejemplo de los ficheros ESRI Shapefile generados por la aplicación web para la publicación interna de datos que se ha desarrollado dentro de este TFG.

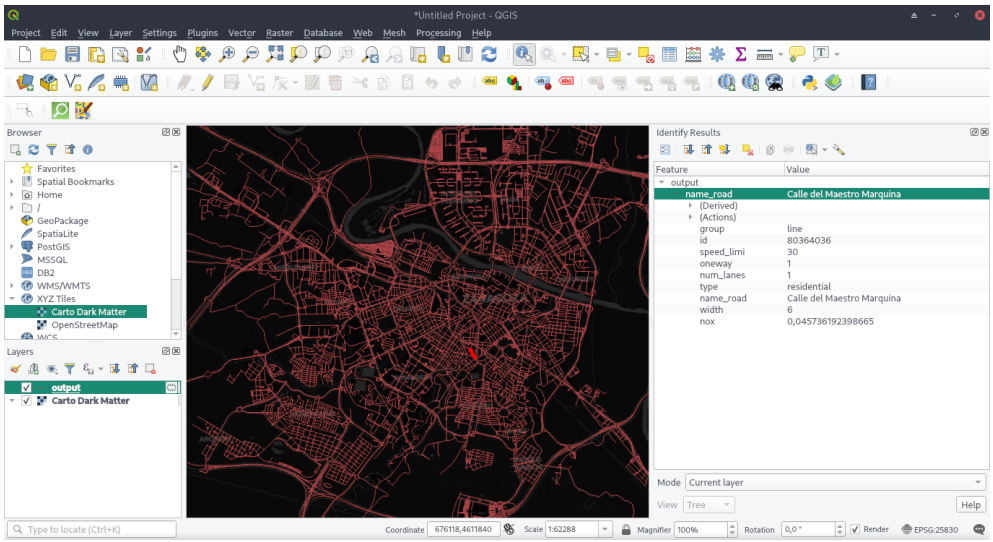


Figura 5.3: Shapefile generado abierto con QGIS. Mapa de fondo: Carto Dark Matter, Colaboradores de OpenStreetMap.

5.2. Publicación de datos abiertos

5.2.1. Descripción del despliegue

La figura 5.4 muestra el despliegue de los portales de Datos Abiertos en las ciudades de Módena (Italia), Santiago de Compostela (España) y Zaragoza (España). La figura también muestra cómo los servidores GeoServer locales son consultados con el software descrito en los pasos 3 y 4 del flujo de trabajo propuesto para insertar los contenidos de los portales de Datos Abiertos locales. Adicionalmente, la figura muestra los portales de Datos Abiertos a nivel regional, nacional y europeo que recolectan los contenidos de los portales de Datos Abiertos locales.

En el caso de Módena, los contenidos de datos abiertos del gobierno municipal de Módena (*Commune di Modena*) se integran directamente en el servidor CKAN mantenido por el gobierno regional de Emilia-Romagna, y posteriormente son recolectados por el portal de Datos Abiertos del Gobierno Italiano (*dati.gov.it*).

En el caso de Santiago de Compostela, el portal de Datos Abiertos está gestionado por el gobierno municipal de Santiago de Compostela (*Concello de Santiago*). Después, los contenidos de este portal son recolectados por el portal de Datos Abiertos del Gobierno de España (*datos.gob.es*).

El caso de Zaragoza es más complejo. Hay un portal de Datos Abiertos

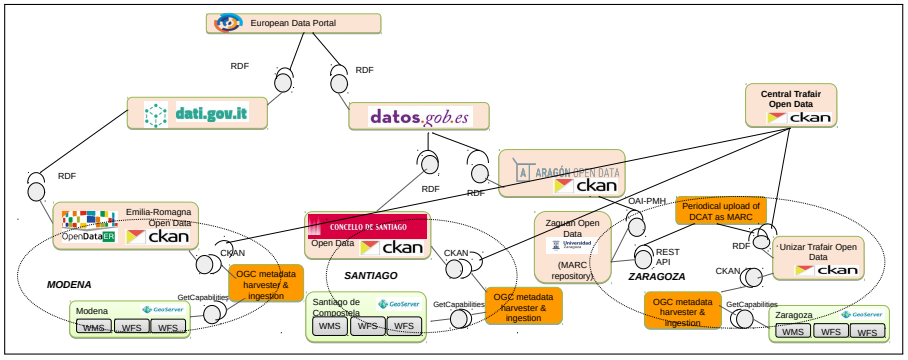


Figura 5.4: Despliegue de servidores de Datos Abiertos en las ciudades de Modena, Santiago y Zaragoza

basado en CKAN mantenido por los investigadores de la Universidad de Zaragoza involucrados en el proyecto TRAFAIR. Sin embargo, los contenidos de Datos Abiertos de la universidad son publicados a través de un repositorio institucional: Zaguan, basado en el esquema de metadatos MARC y accesible a través del protocolo OAI-PMH[50]. En este caso, hemos tenido que implementar un programa específico para cargar periódicamente los metadatos GeoDCAT-AP en formato MARC. Después de ello, estos registros de metadatos son recolectados por el portal del gobierno regional de Aragón (*Aragón Open Data*), y posteriormente por el portal de Datos Abiertos del Gobierno de España.

Por último, debe ser notado que los socios de TRAFAIR han decidido desplegar también un portal de Datos Abiertos central que recolecte los contenidos de los portales individuales en cada ciudad. Este portal de Datos Abiertos centra está también basado en la plataforma software CKAN, y sirve como un catálogo unificado para tener una perspectiva global de todos los contenidos de Datos Abiertos a los que han contribuido los distintos colaboradores del proyecto.

5.2.2. Evaluación de los metadatos

Para evaluar la calidad de los metadatos generados de forma automática gracias a las herramientas desarrolladas en este TFG y que han sido descritas en el capítulo 4, se ha utilizado la metodología de evaluación de calidad de metadatos propuesta por el Portal Europeo de Datos Abiertos y conocida con el acrónimo MQA (Metadata Quality Assesment, MQA) [51].

Esta metodología es la base del cuadro de mandos utilizado en el contexto del Portal Europeo de Datos para proporcionar una visión general de los con-

tenidos recopilados de los diferentes catálogos que contribuyen a este portal europeo. Inspirado en los principios FAIR[52], que proporcionan pautas para mejorar la capacidad de búsqueda, accesibilidad, interoperabilidad y reutilización de activos digitales. MQA propone el uso de 23 métricas clasificadas en cinco dimensiones: facilidad de localización, que verifica la disponibilidad de palabras clave, categorías, información espacial e información temporal; accesibilidad, que verifica la accesibilidad de las URLs de acceso y descarga (incluyendo su existencia); interoperabilidad, que verifica el cumplimiento del modelo de metadatos DCAT-AP y la disponibilidad de formatos conocidos (si es posible, no propietarios y legibles por máquina); reusabilidad, que verifica la descripción de la información de licencia y derechos de acceso, así como los puntos de contacto y editores; y contextualidad, que verifica la disponibilidad de información relacionada con los derechos de distribución, el tamaño del archivo de las distribuciones y las fechas de emisión o modificación. Cada una de esas métricas puede ser asignada a un número máximo de puntos de acuerdo al porcentaje de registros de metadatos que verifican la métrica. El número total de puntos para todas las métricas se usa para clasificar catálogos en los rangos de excelente (351-405 puntos), buena (221-350), suficiente (121-220) o mala (0-120) clasificación.

Dimensión	Indicador/propiedad	Válidos	Población	Porcentaje	Puntos
Facilidad de localización	dcat:keyword	19	19	100	30.0
Facilidad de localización	dcat:theme	19	19	100	30.0
Facilidad de localización	dct:spatial	7	19	36.84	7.368
Facilidad de localización	dct:temporal	4	19	21.05	4.21
Accesibilidad	dcat:accessURL code=200	24	24	100	50.0
Accesibilidad	dcat:downloadURL	0	24	0	0
Accesibilidad	dcat:downloadURL code=200	0	24	0	0
Interoperabilidad	dct:format	24	24	100	20.0
Interoperabilidad	dcat:mediaType	0	24	0	0
Interoperabilidad	dct:format from vocabulary	0	24	0	0
Interoperabilidad	dct:format non-proprietary	5	24	20.83	4.17
Interoperabilidad	dct:format machine-readable	0	24	0	0
Interoperabilidad	DCAT-AP compliance	0	19	0	0
Reusabilidad	dct:license	24	24	100	20.0
Reusabilidad	dct:license from vocabulary	0	24	0	0
Reusabilidad	dct:accessRights	0	19	0	0
Reusabilidad	dct:accessRights from vocabulary	0	19	0	0
Reusabilidad	dcat:contactPoint	19	19	100	20.0
Reusabilidad	dct:publisher	19	19	100	10.0
Contextualidad	dct:rights	24	24	100	5.0
Contextualidad	dcat:byteSize	0	24	0	0
Contextualidad	dct:issued	19	19	100	5.0
Contextualidad	dct:modified	19	19	100	5.0
Puntos totales 210.75					
Clasificación: **Suficiente** (rango 121-220)					

Tabla 5.1: Salida de la herramienta MQA para los conjuntos de datos en CKAN el 20 de mayo de 2020

Como se puede comprobar, la clasificación es suficiente alta. Los campos con porcentaje 0 se deben principalmente a dos motivos: por un lado, algunos campos evaluados por MQA no se consideran en GeoDCAT-AP; por otro

lado, existen algunos campos para cuyos valores se chequea la pertenencia a un vocabulario, donde los vocabularios propuestos por MQA están pensados para recursos de dominios más generales, no específicos del mundo de información geográfica.

Para el resto de campos se comprueba que los metadatos generados cumplen con la metodología de calidad.

6 | Gestión, conclusiones y trabajo futuro

6.1. Gestión

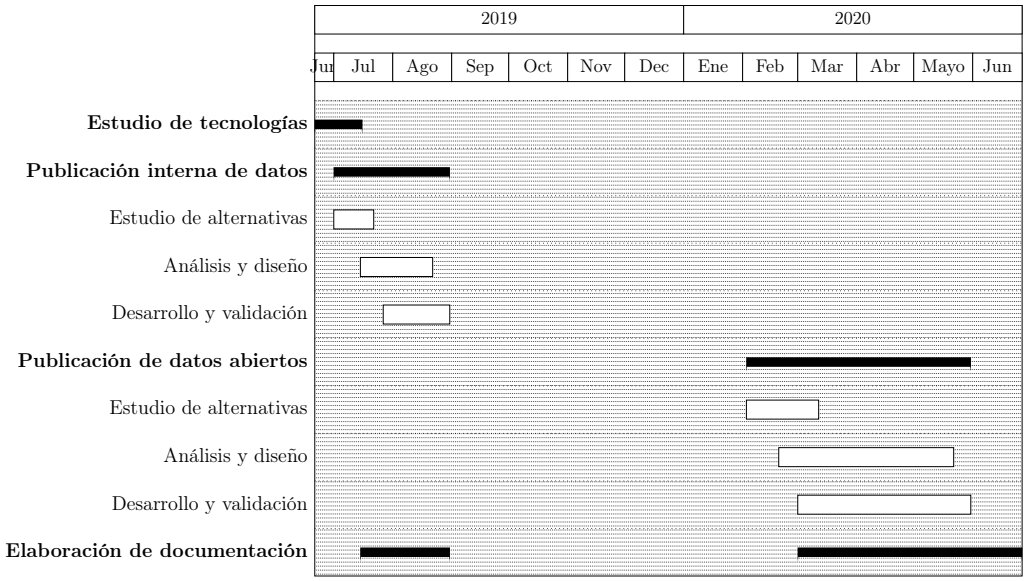


Figura 6.1: Diagrama Gantt del TFG

En el diagrama Gantt (figura 6.1) se puede ver claramente las principales actividades del proyecto y la separación entre las dos fases principales del trabajo. En la recta final (a partir de julio) de las prácticas de investigación que estuve haciendo en TRAFAIR entre marzo y agosto de 2019, realicé la aplicación de publicación interna de datos. También en este periodo se dejó planteada la segunda sección, sobre datos abiertos y que ambas formasen parte del TFG.

Entre septiembre y enero realicé un descanso para centrarme en las asignaturas del primer cuatrimestre y comencé la segunda sección en febrero de 2020, terminando en mayo.

En total se han invertido 445 horas en el proyecto. Las horas invertidas totales en cada actividad han sido las siguientes:

- Estudio de tecnologías: 30 horas
- Publicación interna de datos

  Estudio de alternativas: 25 horas

  Análisis y diseño: 30 horas

  Desarrollo y validación: 55 horas
- Publicación de datos abiertos

  Estudio de alternativas: 50 horas

  Análisis y diseño: 45 horas

  Desarrollo y validación: 130 horas
- Elaboración de documentación: 90 horas

Para la gestión del estado de las tareas se ha utilizado el software Trello[53], distribuyendo las tarjetas de tareas en un panel siguiendo el método Kanban[54] de desarrollo de software. En este método se crean columnas con el estado de cada tarea, y se van moviendo de una a otra dependiendo de lo avanzado que esté en el flujo de trabajo.

La gestión del código con los demás integrantes del proyecto se ha realizado mediante el software de control de versiones Git[55], que permite combinar el trabajo realizado por distintos colaboradores paralelamente en un repositorio, haciendo fácil el manejo de los distintos conflictos que pudiesen surgir. El repositorio del proyecto TRAFAIR se encuentra privado hasta que el proyecto termine, pero se pretende liberar posteriormente.

Para la documentación, concentrada en esta memoria, se ha usado LaTeX[56], un sistema de preparación de documentos que agiliza la redacción de la memoria y permite tener un estilo consistente a lo largo de todo el documento y una correcta gestión de la bibliografía combinándolo con BibTeX[57], un gestor de referencias bibliográficas, y PGF/Ti*k*Z[58] junto con Ti*k*Z-UML, un par de lenguajes para generar diagramas vectoriales en TeX y una librería que usa esos lenguajes para realizar diagramas UML.

El código LaTeX de esta memoria se encuentra disponible para su reutilización en `https://gitlab.com/Robot8A/tfg-trafair-unizar/`.

6.2. Conclusiones

Hemos propuesto un flujo de trabajo para la publicación de Datos Espaciales Abiertos que puede ser personalizado según las necesidades de otros proyectos que trabajen con datos espaciales que necesiten ser accesibles públicamente. Igualmente, hemos demostrado cómo los metadatos GeoDCAT-AP se pueden aplicar en un caso de uso real para describir datos espaciales más específicamente que otros vocabularios de metadatos más generales basados en DCAT.

A partir de este trabajo se ha escrito una publicación[59] enviada como contribución a *The Twelfth International Conference on Advanced Geographic Information Systems, Applications, and Services - GEOProcessing 2020* que se celebrará del 21 al 25 de noviembre del 2020 en Valencia, España.

6.3. Trabajo futuro

Como trabajo futuro, se plantea: integrar el software que se ha desarrollado para la generación y publicación automática de metadatos como un nuevo complemento de CKAN o como una extensión del plugin actual *ckanext-spatial*, proporcionar una interfaz web para el proceso de generación de los metadatos a partir de servicios OGC y el cálculo de los indicadores MQA, y simplificar los pasos de instalación de CKAN y los complementos utilizados con la creación de un docker-compose.

Bibliografía

[1] *TRAFAIR: Understanding traffic flows to improve air quality*. dirección: http://trafair.eu/ (visitado 21-04-2020).

[2] J. R. R. Viqueira, R. Trillo-Lado, S. Villarroya, L. Marrodán, J. M. Cotos, S. Ilarri, J. A. Taboada y E. Torres-Moreno, «Proyecto TRAFAIR: Generación y publicación de datos de calidad del aire en las ciudades de Zaragoza y Santiago de Compostela.», en *Actas de las Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*, 2019.

[3] Wikipedia, *Datos abiertos — Wikipedia, La enciclopedia libre*. dirección: https://es.wikipedia.org/wiki/Datos_abiertos (visitado 02-06-2020).

[4] D. D. Nebert, *Developing Spatial Data Infrastructures: The SDI Cookbook*. 25 de ene. de 2004. dirección: http://www.gsdiassociation.org/images/publications/cookbooks/SDI_Cookbook_GSDI_2004_ver2.pdf (visitado 01-06-2020).

[5] Open Geospatial Consortium, *Catalogue Service*. dirección: https://www.ogc.org/standards/cat (visitado 30-05-2020).

[6] Python Software Foundation, *Python 3.8.3 documentation*. dirección: https://docs.python.org/3/ (visitado 29-04-2020).

[7] G. van Rossum, B. Warsaw y N. Coghlan, «PEP 8 – Style Guide for Python Code», inf. téc., 1 de ago. de 2013. dirección: https://www.python.org/dev/peps/pep-0008/ (visitado 23-04-2020).

[8] Open Source Geospatial Foundation, *GeoServer*. dirección: http://geoserver.org/ (visitado 23-04-2020).

[9] Open Geospatial Consortium, *Web Map Service*. dirección: https://www.ogc.org/standards/wms (visitado 17-05-2020).

[10] ——, *Web Feature Service*. dirección: https://www.ogc.org/standards/wfs (visitado 17-05-2020).

[11] ——, *Web Coverage Service*. dirección: https://www.ogc.org/standards/wcs (visitado 17-05-2020).

[12] *ckan - The open source data portal software*. dirección: https://ckan.org/ (visitado 21-04-2020).

[13] T. Kralidis, A. Tzotsos y R. Clark, *PyCSW - Metadata Publishing Just Got Easier*. dirección: https://pycsw.org/ (visitado 23-04-2020).

[14] Open Knowledge, *ckanext-spatial - Geo related plugins for CKAN*. dirección: https://docs.ckan.org/projects/ckanext-spatial/en/latest/index.html (visitado 30-05-2020).

[15] *GRAL - The Graz Lagrangian Model*. dirección: http://lampz.tugraz.at/~gral/index.php (visitado 06-05-2020).

[16] L. Marrodán Bretón, R. Trillo-Lado, J. Fabra, J. Nogueras-Iso y M. U. Alzueta, «TRAFAIR: Análisis de los flujos de tráfico para mejorar la calidad del aire urbano», en *Actas de la VIII Jornada de Jóvenes Investigadores del I3A, Zaragoza, Spain, 6 June 2019*, 2019.

[17] D. Sáez García, «Desarrollo e implementación de herramientas de monitorización y simulación de tráfico basadas en datos abiertos», Trabajo Fin de Grado, Universidad de Zaragoza, 2019.

[18] Oficina de Publicaciones de la Unión Europea, *Portal Europeo de Datos*. dirección: https://www.europeandataportal.eu/es (visitado 21-04-2020).

[19] Comisión Europea, *GeoDCAT Application profile for data portals in Europe, GeoDCAT-AP v1.0.1*, https://joinup.ec.europa.eu/release/geodcat-ap/101, 2016.

[20] ——, *DCAT Application Profile for data portals in Europe, DCAT-AP v2.0.0*, https://joinup.ec.europa.eu/solution/dcat-application-profile-data-portals-europe/release/200, 2019.

[21] W3C, *Data Catalog Vocabulary (DCAT). W3C Working Draft,12 March 2013*, http://www.w3.org/TR/2013/WD-vocab-dcat-20130312/, 2013.

[22] Comisión Europea, «REGLAMENTO (CE) Nº 1205/2008 DE LA COMISIÓN de 3 de diciembre de 2008 por el que se ejecuta la Directiva 2007/2/CE del Parlamento Europeo y del Consejo en lo que se refiere a los metadatos», inf. téc., 3 de dic. de 2008. dirección: https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2008:326:0012:0030:ES:PDF (visitado 13-05-2020).

[23] Open Knowledge, *ckan/ckanext-dcat - CKAN ♡ DCAT - Github*. dirección: `https://github.com/ckan/ckanext-dcat` (visitado 26-05-2020).

[24] Civic Actions, *DKAN Open Data Platform | DKAN Open Data Platform*. dirección: `https://getdkan.org/`.

[25] D. Oettl, «Documentation of the Lagrangian Particle Model - GRAL (Graz Lagrangian Model)», Government of Styria - Department 15 Energy, Housing, Technology, Graz, AT, inf. téc., ver. 19.1, ene. de 2019.

[26] The Linux Foundation®, *Home - OpenAPI Initiative*. dirección: `https://www.openapis.org/` (visitado 04-05-2020).

[27] Zalando SE, *Welcome to Connexion's documentation!* Dirección: `https://connexion.readthedocs.io/en/latest/` (visitado 04-05-2020).

[28] *OpenStreetMap*. dirección: `https://www.openstreetmap.org/` (visitado 12-06-2020).

[29] *Overpass API - OpenStreetMap Wiki*. dirección: `https://wiki.openstreetmap.org/wiki/Overpass_API` (visitado 12-06-2020).

[30] The PostgreSQL Global Development Group, *PostgreSQL: The World's Most Advanced Open Source Relational Database*. dirección: `https://www.postgresql.org/` (visitado 21-06-2020).

[31] *PostGIS - Spatial and Geographic objects for PostgreSQL*. dirección: `https://postgis.net/` (visitado 21-06-2020).

[32] *GDAL/OGR Python API*. dirección: `https://gdal.org/python/` (visitado 21-06-2020).

[33] E. Rouault y F. Warmerdam, *ogr2ogr.py*. dirección: `https://svn.osgeo.org/gdal/trunk/gdal/swig/python/samples/ogr2ogr.py` (visitado 21-06-2020).

[34] Contribuidores de OpenStreetMap, *Key:highway - OpenStreetMap Wiki*. dirección: `https://wiki.openstreetmap.org/wiki/Key:highway#Values` (visitado 22-06-2020).

[35] *European Petroleum Survey Group projection 25830: ETRS89 / UTM zone 30N – Spatial Reference*. dirección: `https://spatialreference.org/ref/epsg/25830/` (visitado 06-05-2020).

[36] *European Petroleum Survey Group projection 4326: WGS 84*. dirección: `https://spatialreference.org/ref/epsg/4326/` (visitado 06-05-2020).

[37] Xunta de Galicia, *Servidor THREDDS de MeteoGalicia*. dirección: https://www.meteogalicia.gal/web/modelos/threddsIndex.action?request_locale=es (visitado 21-06-2020).

[38] ISO, «ISO 19115-1:2014. Geographic information - Metadata - Part 1: Fundamentals», Organización Internacional de Normalización, Ginebra, CH, inf. téc., 2014.

[39] INSPIRE MIG, «Technical Guidelines for implementing dataset and service metadata based on ISO/TS 19139:2007», INSPIRE Maintenance e Implementation Group (MIG), INSPIRE Maintenance and Implementation Group (MIG). Version 2.0.1, 2 de mar. de 2017. dirección: http://inspire.ec.europa.eu/id/document/tg/metadata-iso19139 (visitado 27-05-2020).

[40] *GeoServer API Docs - API for Features*. dirección: https://docs.geoserver.org/latest/en/api/#/latest/en/api/1.0.0/featuretypes.yaml (visitado 27-05-2020).

[41] *GeoServer API Docs - API for Coverages*. dirección: https://docs.geoserver.org/latest/en/api/#/latest/en/api/1.0.0/coveragetypes.yaml (visitado 27-05-2020).

[42] T. Kralidis, *OWSLib 0.19.2 documentation*. dirección: https://geopython.github.io/OWSLib/ (visitado 28-05-2020).

[43] Open Knowledge International and contributors, *API guide - CKAN 2.8.2 documentation*. dirección: https://docs.ckan.org/en/2.8/api/index.html (visitado 28-05-2020).

[44] Universidad de Zaragoza, *Zaguan - Repositorio Institucional de Documentos*. dirección: http://zaguan.unizar.es/ (visitado 21-04-2020).

[45] CERN & contributors, *Invenio - Powering Open Science*. dirección: https://invenio-software.org/ (visitado 21-04-2020).

[46] Biblioteca del Congreso de los EEUU - Oficina de desarrollo de Redes y Normas MARC, *Normas MARC*. dirección: https://www.loc.gov/marc/marcspa.html (visitado 21-04-2020).

[47] Gobierno de Aragón, *Aragón Open Data*. dirección: https://opendata.aragon.es/ (visitado 21-04-2020).

[48] Gobierno de España, Vicepresidencia Tercera del Gobierno, Ministerio de Asuntos Económicos y Transformación Digital, *datos.gob.es, reutiliza la información pública*. dirección: https://datos.gob.es/ (visitado 21-04-2020).

[49] Open Source Geospatial Foundation, *Home - GeoNetwork opensource*. dirección: `https://www.geonetwork-opensource.org/`.

[50] C. Lagoze, H. Van de Sompel, M. Nelson y S. Warner, «The Open Archives Initiative Protocol for Metadata Harvesting», inf. téc., ver. 2.0, 8 de ene. de 2015. dirección: `https://www.openarchives.org/OAI/openarchivesprotocol.html` (visitado 28-05-2020).

[51] Oficina de Publicaciones de la Unión Europea, *Metodología de evaluación de la calidad de los metadatos. Método empleado por EDP para medir la calidad de los datos recolectados*. dirección: `https://www.europeandataportal.eu/mqa/methodology?locale=es` (visitado 23-05-2020).

[52] M. D. Wilkinson et al., «The FAIR Guiding Principles for scientific data management and stewardship», *Scientific data*, vol. 3, 2016.

[53] *Trello*. dirección: `https://trello.com/home` (visitado 02-06-2020).

[54] Wikipedia, *Kanban (desarrollo) — Wikipedia, La enciclopedia libre*. dirección: `https://es.wikipedia.org/wiki/Kanban_(desarrollo)` (visitado 02-06-2020).

[55] *git –distributed-even-if-your-workflow-isnt*. dirección: `https://git-scm.com/` (visitado 08-06-2020).

[56] *LaTeX - A document preparation system*. dirección: `https://www.latex-project.org/` (visitado 23-04-2020).

[57] *BibTeX - TeX Users Group*. dirección: `https://tug.org/bibtex/` (visitado 23-04-2020).

[58] T. Tantau, «The Ti*k*Z and PGF Packages. Manual for version 3.1.5b», inf. téc., 8 de ene. de 2020. dirección: `http://ftp.ntua.gr/mirror/ctan/graphics/pgf/base/doc/pgfmanual.pdf` (visitado 28-04-2020).

[59] J. Nogueras-Iso, H. Ochoa-Ortiz, M. Á. Jañez, J. R. R. Viqueira, L. Po y R. Trillo-Lado, «Automatic publication of Open Data from OGC services: the use case of TRAFAIR project», Enviado a *The Twelfth International Conference on Advanced Geographic Information Systems, Applications, and Services - GEOProcessing 2020*, 2020.

Índice de figuras

Índice de tablas

Índice de códigos

Anexos

A. Manual de instalación de CKAN

Se utiliza Ubuntu 18.04 junto con docker-compose.

A.1. Entorno, instalar docker

Las instrucciones para instalar docker están disponibles en: `https://docs.docker.com/install/linux/docker-ce/ubuntu/`

```
sudo apt-get remove docker docker-engine docker.io containerd runc
sudo apt-get update
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo apt-key fingerprint 0EBFCD88
sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
sudo docker run hello-world
```

Algunas acciones post-instalación están explicadas en `https://docs.docker.com/install/linux/linux-postinstall/`

```
sudo groupadd docker
sudo usermod -aG docker USER
```

Cerrar sesión

```
newgrp docker
docker run hello-world
```

Configurar Docker para comenzar al inicio del sistema

```
$ sudo systemctl enable docker
```

Para desactivar esta característica, usar disable.

```
$ sudo systemctl disable docker
```

Instalación de Docker-compose de acuerdo con las pautas dadas en `https://docs.docker.com/compose/install/`

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.25.0/docker-compose-$(uname -s)-$(uname -m)" -o /
sudo chmod +x /usr/local/bin/docker-compose
```

A.2. Crear imagen de CKAN

Clonar proyecto

```
cd /path/to/my/projects
git clone https://github.com/ckan/ckan.git
```

Doescargar una versión estable (2.8.3)

```
cd ckan
git checkout tags/ckan-2.8.3
Build image
cd contrib/docker
cp .env.template .env
```

Editar .env si se quiere redireccionar CKAN_SITE_URL a la IP de tu host (Por defecto, CKAN funcionará en localhost:5000)

```
docker-compose up -d --build
```

En el caso de que algún problema suceda, CKAN puede ser reiniciado y revisar los logs:

```
docker-compose restart ckan
docker ps | grep ckan
docker-compose logs -f ckan
```

Crear variables para acceder a los volúmenes (como paso previo, instalar jq)

```
sudo snap install jq  # version 1.5+dfsg-1

export VOL_CKAN_HOME=`docker volume inspect docker_ckan_home | jq -r -c '.[] | .Mountpoint'`
echo $VOL_CKAN_HOME

export VOL_CKAN_CONFIG=`docker volume inspect docker_ckan_config | jq -r -c '.[] | .Mountpoint'`
echo $VOL_CKAN_CONFIG

export VOL_CKAN_STORAGE=`docker volume inspect docker_ckan_storage | jq -r -c '.[] | .Mountpoint'`
echo $VOL_CKAN_STORAGE
```

Las variables pueden añadirse a `$HOME/.bashrc` para no tener que ejecutar el código export cada vez que se necesiten.

A.3. Datastore, datapusher

El siguiente script crea la base de datos datastore y el usuario de solo lectura datastore en el contenedor db.

```
docker exec -it db sh /docker-entrypoint-initdb.d/00_create_datastore.sh
```

El segundo script es la salida de `paster ckan set-permissions`. Pero, como esta salida puede cambiar en futuras versiones de CKAN, ponemos los permisos directamente. El efecto de estos scripts es persistente en el volumen docker_pg_data.

```
docker exec ckan /usr/local/bin/ckan-paster --plugin=ckan datastore set-permissions \
    -c /etc/ckan/production.ini | docker exec -i db psql -U ckan
```

Editar production.ini

```
sudo vim $VOL_CKAN_CONFIG/production.ini
```

para actualizar las siguientes variables:

```
ckan.plugins = stats text_view image_view recline_view datastore datapusher

ckan.datapusher.formats = csv xls xlsx tsv application/csv application/vnd.ms-excel
    application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
```

Si todo está bien, el sitio CKAN debería mostrar las páginas `http://localhost:5000/api/3/action/datastore_search?resource_id=_table_metadata` y `http://localhost:5000/api/3/action/datastore_search?resource_id=_table_metadata`

A.4. Crear usuario de administración CKAN

El siguiente comando permite introducir el e-mail y la contraseña del administrador del sitio CKAN:

```
docker exec -it ckan /usr/local/bin/ckan-paster --plugin=ckan \
    sysadmin -c /etc/ckan/production.ini add admin
```

A.5. Migrar datos

Si tuviésemos una instancia anterior de CKAN y se desea migrar los archivos, es necesario ejecutar el siguiente comando:

```
sudo rsync -Pavvr USER@SOURCE_CKAN:/path/to/files/ $VOL_CKAN_STORAGE
There is also a guideline to export users and groups.
```

A.6. Añadir extensión ckanext-harvest

Esta extensión permite la recolecta de datos (harvest) de otros sitios CKAN. Información sobre la extensión en `https://github.com/ckan/ckanext-harvest`.

Entrar al contenedor ckan que está corriendo:

```
docker exec -it ckan /bin/bash -c "export TERM=xterm; exec bash"
```

Dentro del contenedor ckan

```
source $CKAN_VENV/bin/activate && cd $CKAN_VENV/src/
pip install -e git+https://github.com/ckan/ckanext-harvest.git#egg=ckanext-harvest
cd ckanext-harvest
pip install -r pip-requirements.txt
exit
```

En el anfitrión, editar `$VOL_CKAN_CONFIG/production.ini` con los siguientes valores:

```
ckan.plugins = ... harvest ckan_harvester
ckan.harvest.mq.type = redis
ckan.harvest.mq.hostname = redis
```

Inicializar complemento y reiniciar CKAN:

```
docker exec -it ckan /usr/local/bin/ckan-paster --plugin=ckanext-harvest harvester initdb \
    --config=/etc/ckan/production.ini
docker-compose restart ckan
```

A.7. Añadir extensión ckanext-spatial

Esta extensión da soporte a campos especiales sobre información espacial, indexados en esos campos de información espacial y recolecta de catálogos CSW. Información sobre la extensión en `https://github.com/ckan/ckanext-spatial`.

Entrar al contenedor ckan que está corriendo:

```
docker exec -it ckan /bin/bash -c "export TERM=xterm; exec bash"
```

Dentro del contenedor ckan

```
source $CKAN_VENV/bin/activate && cd $CKAN_VENV/src/
git clone https://github.com/ckan/ckanext-spatial.git
cd ckanext-spatial
pip install -r pip-requirements.txt
python setup.py install && python setup.py develop
exit
```

En el anfitrión

```
docker exec -it db psql -U ckan -f /docker-entrypoint-initdb.d/20_postgis_permissions.sql
```

En el anfitrión, editar `$VOL_CKAN_CONFIG/production.ini` con los siguientes valores:

```
ckan.plugins = ... spatial_metadata spatial_query spatial_harvest_metadata_api
    csw_harvester waf_harvester doc_harvester
ckan.spatial.srid = 4326
ckanext.spatial.search_backend = solr
```

Inicializar complemento y reiniciar CKAN:

```
docker exec -it ckan /usr/local/bin/ckan-paster --plugin=ckanext-spatial spatial initdb -c /etc/ckan/production.ini
docker-compose restart ckan
```

Explicación de recolectores:

- csw_harvester: servidor CSW
- waf_harvester: WAF (Web Accessible Folder): Una página de índice accesible online con enlaces a documentos de metadatos
- doc_harvester: Un documento único sobre metadatos accesible online.

Para activar pyCSW:

Incluir un fichero con el siguiente código en entrypoint del contenedor db y correrlo:

```
#!/bin/bash
set -e

psql -v ON_ERROR_STOP=1 --username "$POSTGRES_USER" <<-EOSQL
    CREATE ROLE pycsw_ro NOSUPERUSER NOCREATEDB NOCREATEROLE
        LOGIN PASSWORD '$POSTGRES_PASSWORD';
    CREATE DATABASE pycsw OWNER ckan ENCODING 'utf-8';
    GRANT ALL PRIVILEGES ON DATABASE pycsw TO ckan;
EOSQL
```

Añadir en ports de ckan dentro del fichero docker-compose.yml

```
- "0.0.0.0:8000:8000"
- "0.0.0.0:80:80"
```

Entrar en contenedor, activar virtualenv

```
git clone https://github.com/geopython/pycsw.git
cd pycsw
git checkout 1.10.4
pip install -e .
python setup.py build
python setup.py install
cp default-sample.cfg default.cfg
```

Modificar default.cfg (ya sea desde dentro del venv o desde fuera en la ruta `$VOL_CKAN_HOME/venv/src/pycsw/default.cfg`) para cambiar:

```
[server]
home=/usr/lib/ckan/venv/src/pycsw

...

[repository]
database=postgresql://ckan:ckan@db/pycsw
```

y los detalles extra que se quieran cambiar.

dentro del virtualenv:

```
cd ../ckanext-spatial
paster ckan-pycsw setup -p ../pycsw/default.cfg
cd ../pycsw
python csw.wsgi
```

Esto empezará pyCSSW en http://localhost:8000. Al visitar la siguiente url debería devolver el fichero de Capabilities:

`http://localhost:8000/?service=CSW&version=2.0.2&request=GetCapabilities`

Cargar los conjuntos de datos de CKAN en pyCSW. Usaremos el comando ckan-pycsw para esto:

```
cd ../ckanext-spatial
paster ckan-pycsw load -p ../pycsw/default.cfg -u http://localhost:5000
```

A.8. Añadir extensión ckanext-geoview

Esta extensión incluye visores para algunos formatos espaciales (geojson, shapefile) y servicios (WMS, WMTS, ...). Información sobre la extensión en `https://github.com/ckan/ckanext-geoview`.

Entrar al contenedor ckan que está corriendo:

```
docker exec -it ckan /bin/bash -c "export TERM=xterm; exec bash"
```

Dentro del contenedor ckan

```
source $CKAN_VENV/bin/activate && cd $CKAN_VENV/src/
git clone https://github.com/ckan/ckanext-geoview.git
cd ckanext-geoview
pip install -r pip-requirements.txt
python setup.py install
python setup.py develop
cd ..
exit
```

En el anfitrión, editar `$VOL_CKAN_CONFIG/production.ini` con los siguientes valores:

```
ckan.plugins = ... resource_proxy geo_view geojson_view wmts_view
ckan.views.default_views = ... wmts_view geojson_view geo_view
```

Reiniciar CKAN:

```
docker-compose restart ckan
```

A.9. Añadir extensión ckanext-dcat

Esta extensión permite la generación de metadatos DCAT RDF. El mapeo entre campos CKAN y propiedades DCAT RDF está explicado en `https://github.com/ckan/ckanext-dcat#rdf-dcat-to-ckan-dataset-mapping`. Información sobre la extensión en `https://github.com/ckan/ckanext-dcat`.

Entrar al contenedor ckan que está corriendo:

```
docker exec -it ckan /bin/bash -c "export TERM=xterm; exec bash"
```

Dentro del contenedor ckan

```
source $CKAN_VENV/bin/activate && cd $CKAN_VENV/src/
pip install -e git+https://github.com/ckan/ckanext-dcat.git#egg=ckanext-dcat
cd ckanext-dcat
pip install -r requirements.txt
exit
```

En el anfitrión, editar `$VOL_CKAN_CONFIG/production.ini` con los siguientes valores:

```
ckan.plugins = ... dcat dcat_rdf_harvester dcat_json_harvester dcat_json_interface structured_data
```

Reiniciar CKAN:

```
docker-compose restart ckan
```

A.10. Cómo parar y volver a lanzar CKAN (y componentes asociados)

Para parar CKAN si se va a apagar la máquina, hay que realizar lo siguiente:

```
cd /path/to/my/ckan/folder/contrib/docker
docker-compose stop
```

Para volver a iniciar CKAN si la máquina es reiniciada, hacer lo siguiente:

```
cd /path/to/my/ckan/folder/contrib/docker
docker-compose up -d
```

A.11. Pequeños pasos para verificar la correcta instalación

- Abrir el portal CKAN desde un navegador web (`http://localhost:5000` o la dirección que se ha indicado en CKAN_SITE_URL).
- Iniciar sesión como administrador o usuario con suficientes privilegios como para crear organizaciones y conjuntos de datos.
- Crear una organización.

- Crear un conjunto de datos con un nombre reconocible (por ejemplo example_dataset) enlazado a la organización que acabamos de crear.
- Exportar el conjunto de datos anterior como RDF gracias a la extensión ckan-dcat. Simplemente hace falta escribir la extensión .rdf al final de la URL del conjunto de datos (ejemplo: `http://localhost:5000/dataset/example_dataset.rdf`)

B. Ejemplo de un registro de metadatos en sus diferentes fases

A continuación podemos ver las diferentes fases por las que un registro de ejemplo pasa y las transformaciones de metadatos que ocurren.

La figura B.1 muestra una lista de capas servidas por GeoServer.

El código B.1 contiene el extracto de la llamada GetCapabilities correspondiente al registro de ejemplo que vamos a seguir en este anexo del WMS servido por GeoServer.

La figura B.2 muestra el detalle del conjunto de datos subido en el servidor CKAN local por la pasarela OGC-CKAN y el código B.2 el registro GeoDCAT-AP RDF en ese mismo instante.

En el código B.3 enseña la transformación que el anterior registro ha sufrido tras pasar por la pasarela CKAN-Zaguan y en la figura B.3 se ve ya subido al repositorio institucional.

En la figura B.4 se observa el conjunto de datos después de haber sido recolectado desde Zaguan y en B.4 comprobamos el registro GeoDCAT-AP RDF en ese instante.

Por último, el conjunto de datos es recolectado por otros dos servidores CKAN de nivel superior: datos.gob.es (figura B.5) y finalmente el Portal Europeo de Datos (figura B.6).

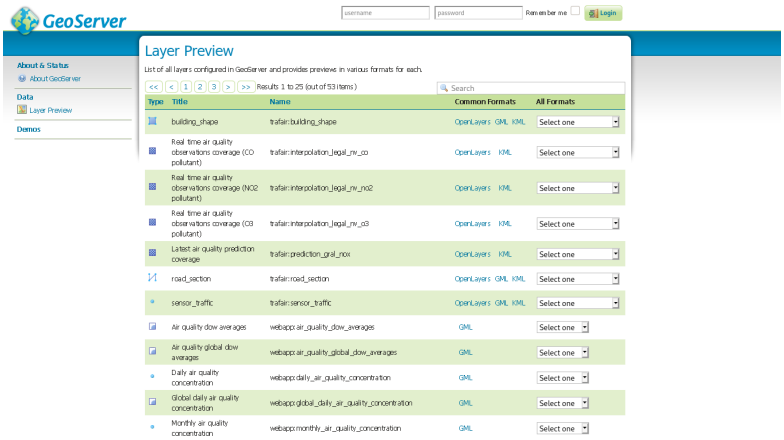


Figura B.1: Lista de capas servidas por GeoServer

```
<Layer queryable="1" opaque="0">
<Name>prediction_gral_nox</Name>
<Title>Latest air quality prediction coverage</Title>
<Abstract>
This dataset provides the latest air quality prediction generated by GRAL
     model, one time instant for each of the following 48 hours.
</Abstract>
```

```
<KeywordList>
<Keyword>prediction_gral_nox</Keyword>
<Keyword>WCS</Keyword>
<Keyword>ImageMosaic</Keyword>
</KeywordList>
<CRS>EPSG:32630</CRS>
<CRS>CRS:84</CRS>
<EX_GeographicBoundingBox>
<westBoundLongitude>-0.9372715072413826</westBoundLongitude>
<eastBoundLongitude>-0.8433593953634075</eastBoundLongitude>
<southBoundLatitude>41.61805345955938</southBoundLatitude>
<northBoundLatitude>41.69206727486579</northBoundLatitude>
</EX_GeographicBoundingBox>
<BoundingBox CRS="CRS:84" minx="-0.9372715072413826" miny
     ="41.61805345955938" maxx="-0.8433593953634075" maxy
     ="41.69206727486579"/>
<BoundingBox CRS="EPSG:32630" minx="671848.0" miny="4609612.0" maxx
     ="679476.0" maxy="4617648.0"/>
<Dimension name="time" default="2020-06-21T23:00:00Z" units="ISO8601">
2020-01-22T00:00:00.000Z,2020-01-22T01:00:00.000Z,2020-01-22T02:00:00.000Z
     ,2020-01-22T03:00:00.000Z,2020-01-22T04:00:00.000Z...
</Dimension>
<Style>
<Name>mobile:nox_gral</Name>
<Title>mobile:nox_gral</Title>
<LegendURL width="81" height="213">
<Format>image/png</Format>
<OnlineResource xlink:type="simple" xlink:href="http://atila.unizar.es:8081/
     geoserver/trafair/ows?service=WMS&request=GetLegendGraphic&format=image
     %2Fpng&width=20&height=20&layer=prediction_gral_nox"/>
</LegendURL>
</Style>
<Style>
<Name>EU_NO2_style_custom</Name>
<Title>EU_NO2_style_custom</Title>
<LegendURL width="46" height="152">
<Format>image/png</Format>
<OnlineResource xlink:type="simple" xlink:href="http://atila.unizar.es:8081/
     geoserver/trafair/ows?service=WMS&request=GetLegendGraphic&format=image
     %2Fpng&width=20&height=20&layer=prediction_gral_nox&style=
     EU_NO2_style_custom"/>
</LegendURL>
</Style>
<Style>
<Name>EU_NO2_style</Name>
<Title>EU_NO2_style</Title>
<LegendURL width="54" height="152">
<Format>image/png</Format>
<OnlineResource xlink:type="simple" xlink:href="http://atila.unizar.es:8081/
     geoserver/trafair/ows?service=WMS&request=GetLegendGraphic&format=image
     %2Fpng&width=20&height=20&layer=prediction_gral_nox&style=EU_NO2_style
     "/>
</LegendURL>
</Style>
</Layer>
```

Código B.1: Extracto de la llamada GetCapabilities del WMS servido por GeoServer mostrando los metadatos de un único conjunto de datos

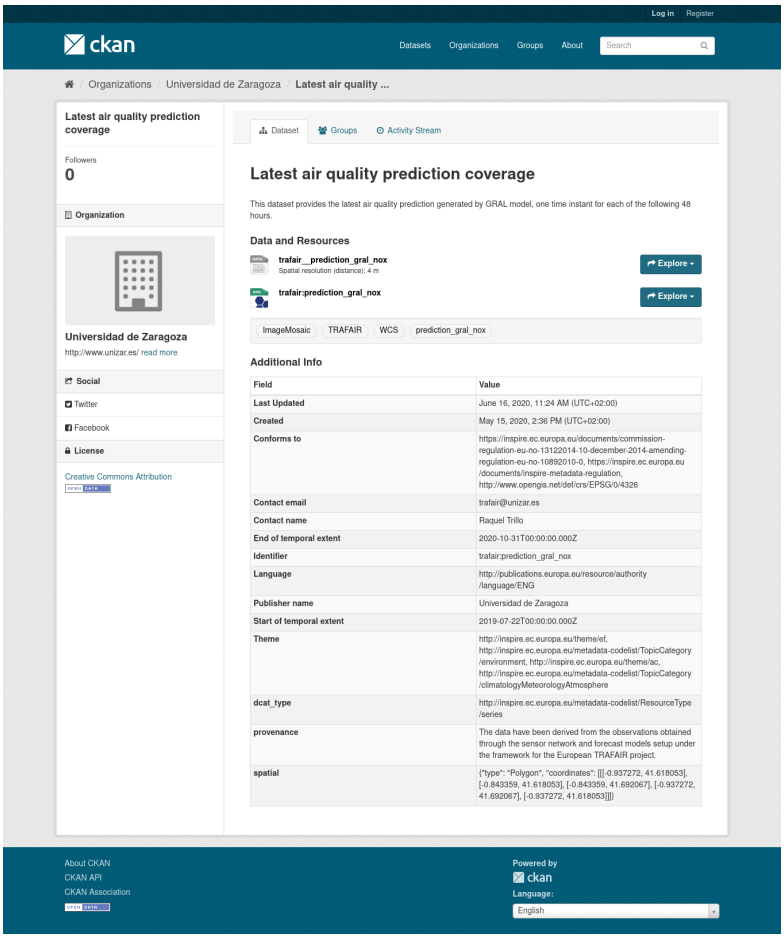


Figura B.2: Detalle de un conjunto de datos subido en el servidor CKAN local

```
<rdf:RDF>
<dcat:Dataset rdf:about="http://atila.unizar.es:3394/dataset/dbb3799e-dc88-4
    e8f-995b-7d8e78779461">
<dct:conformsTo>
https://inspire.ec.europa.eu/documents/inspire-metadata-regulation
</dct:conformsTo>
<dcat:keyword>prediction_gral_nox</dcat:keyword>
<dcat:contactPoint>
<vcard:Organization rdf:nodeID="N5bde12e2e2ed41438d0af18bfaf6e510">
<vcard:hasEmail rdf:resource="mailto:trafair@unizar.es"/>
<vcard:fn>Raquel Trillo</vcard:fn>
</vcard:Organization>
</dcat:contactPoint>
<dcat:theme rdf:resource="http://inspire.ec.europa.eu/theme/ef"/>
<dct:spatial>
<dct:Location rdf:nodeID="N0002583774a44c748649e37f3a2768ba">
<locn:geometry rdf:datatype="https://www.iana.org/assignments/media-types/
    application/vnd.geo+json">
{"type": "Polygon", "coordinates": [[[-0.937272, 41.618053], [-0.843359,
    41.618053], [-0.843359, 41.692067], [-0.937272, 41.692067], [-0.937272,
     41.618053]]]}
</locn:geometry>
<locn:geometry rdf:datatype="http://www.opengis.net/ont/geosparql#wktLiteral
    ">
POLYGON ((-0.9373 41.6181, -0.8434 41.6181, -0.8434 41.6921, -0.9373
    41.6921, -0.9373 41.6181))
```

```
</locn:geometry>
</dct:Location>
</dct:spatial>
<dcat:keyword>TRAFAIR</dcat:keyword>
<dct:temporal>
<dct:PeriodOfTime rdf:nodeID="N2f1739e8a0214249af83717974983b97">
<schema:endDate rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime
      ">2020-10-31T00:00:00+00:00</schema:endDate>
<schema:startDate rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime
      ">2019-07-22T00:00:00+00:00</schema:startDate>
</dct:PeriodOfTime>
</dct:temporal>
<dct:issued rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime
      ">2020-05-15T12:36:44.649560</dct:issued>
<dcat:keyword>ImageMosaic</dcat:keyword>
<dct:title>Latest air quality prediction coverage</dct:title>
<dcat:distribution>
<dcat:Distribution rdf:about="http://atila.unizar.es:3394/dataset/dbb3799e-
      dc88-4e8f-995b-7d8e78779461/resource/01f6dfe6-521c-4f46-af56-89
      d2a634b2b1">
<dct:rights rdf:resource="http://inspire.ec.europa.eu/metadata-codelist/
      LimitationsOnPublicAccess/noLimitations"/>
<dcat:accessURL rdf:resource="http://atila.unizar.es:8081/geoserver/ows?
      service=WCS&request=GetCapabilities"/>
<dct:title>trafair__prediction_gral_nox</dct:title>
<dct:format>WCS</dct:format>
<dct:license rdf:resource="https://creativecommons.org/licenses/by-nc
      /4.0/"/>
<dct:description>Spatial resolution (distance): 4 m</dct:description>
</dcat:Distribution>
</dcat:distribution>
<dcat:keyword>WCS</dcat:keyword>
<dct:identifier>trafair:prediction_gral_nox</dct:identifier>
<dct:conformsTo> http://www.opengis.net/def/crs/EPSG/0/4326</dct:conformsTo>
<dct:modified rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime
      ">2020-06-16T09:24:50.249824</dct:modified>
<dcat:theme rdf:resource="http://inspire.ec.europa.eu/metadata-codelist/
      TopicCategory/environment"/>
<dct:type>
http://inspire.ec.europa.eu/metadata-codelist/ResourceType/series
</dct:type>
<dct:publisher>
<foaf:Organization rdf:about="http://atila.unizar.es:3394/organization/
      d8653565-dacb-441b-93ef-de407462f11e">
<foaf:name>Universidad de Zaragoza</foaf:name>
</foaf:Organization>
</dct:publisher>
<dct:description>
This dataset provides the latest air quality prediction generated by GRAL
      model, one time instant for each of the following 48 hours.
</dct:description>
<dct:provenance>
The data have been derived from the observations obtained through the sensor
       network and forecast models setup under the framework for the European
       TRAFAIR project.
</dct:provenance>
<dcat:theme rdf:resource="http://inspire.ec.europa.eu/theme/ac"/>
<dct:conformsTo>
https://inspire.ec.europa.eu/documents/commission-regulation-eu-no
      -13122014-10-december-2014-amending-regulation-eu-no-10892010-0
</dct:conformsTo>
<dcat:distribution>
<dcat:Distribution rdf:about="http://atila.unizar.es:3394/dataset/dbb3799e-
      dc88-4e8f-995b-7d8e78779461/resource/7c58c631-ffd7-4d8a-9421-
      fb8f87b06cb0">
<dct:license rdf:resource="https://creativecommons.org/licenses/by-nc
      /4.0/"/>
<dcat:accessURL rdf:resource="http://atila.unizar.es:8081/geoserver/ows?
      service=wms&request=GetCapabilities"/>
<dct:format>WMS</dct:format>
<dct:rights rdf:resource="http://inspire.ec.europa.eu/metadata-codelist/
      LimitationsOnPublicAccess/noLimitations"/>
<dct:title>trafair:prediction_gral_nox</dct:title>
</dcat:Distribution>
</dcat:distribution>
<dct:language rdf:resource="http://publications.europa.eu/resource/authority
      /language/ENG"/>
<dcat:theme rdf:resource="http://inspire.ec.europa.eu/metadata-codelist/
      TopicCategory/climatologyMeteorologyAtmosphere"/>
</dcat:Dataset>
```

```
</rdf:RDF>
```

Código B.2: GeoDCAT-AP RDF de un conjunto de datos servido desde el servidor CKAN local

```
<record>
<datafield ind1=" " ind2=" " tag="037">
<subfield code="a">TRAFAIR-2020-007</subfield>
</datafield>
<datafield ind1=" " ind2=" " tag="970">
<subfield code="a">TRAFAIR-2020-007</subfield>
</datafield>
<datafield ind1=" " ind2=" " tag="245">
<subfield code="a">Latest air quality prediction coverage</subfield>
</datafield>
<datafield ind1="3" ind2=" " tag="520">
<subfield code="a">
This dataset provides the latest air quality prediction generated by GRAL
     model, one time instant for each of the following 48 hours.
</subfield>
</datafield>
<datafield ind1="1" ind2=" " tag="653">
<subfield code="a">TRAFAIR</subfield>
</datafield>
<datafield ind1="1" ind2=" " tag="653">
<subfield code="a">prediction_gral_nox</subfield>
</datafield>
<datafield ind1="1" ind2=" " tag="653">
<subfield code="a">WCS</subfield>
</datafield>
<datafield ind1="1" ind2=" " tag="653">
<subfield code="a">ImageMosaic</subfield>
</datafield>
<datafield ind1=" " ind2=" " tag="260">
<subfield code="b">Universidad de Zaragoza</subfield>
<subfield code="c">15/05/2020</subfield>
</datafield>
<datafield ind1=" " ind2=" " tag="700">
<subfield code="a">Raquel Trillo</subfield>
</datafield>
<datafield ind1=" " ind2=" " tag="980">
<subfield code="a">OPENDATA</subfield>
<subfield code="b">RESEARCH</subfield>
<subfield code="c">TRAFAIR</subfield>
</datafield>
<datafield ind1=" " ind2="7" tag="655">
<subfield code="2">inspire</subfield>
<subfield code="a">
http://inspire.ec.europa.eu/metadata-codelist/ResourceType/series
</subfield>
</datafield>
<datafield ind1="4" ind2=" " tag="856">
<subfield code="n">trafair__prediction_gral_nox</subfield>
<subfield code="u">
http://atila.unizar.es:8081/geoserver/ows?service=WCS&request=
     GetCapabilities
</subfield>
<subfield code="y">WCS</subfield>
</datafield>
<datafield ind1=" " ind2=" " tag="540">
<subfield code="u">https://creativecommons.org/licenses/by-nc/4.0/</subfield
     >
<subfield code="2">cc</subfield>
<subfield code="a">by-nc</subfield>
<subfield code="b">Creative Commons</subfield>
<subfield code="c">4.0</subfield>
<subfield code="f">CC BY-NC 4.0</subfield>
<subfield code="9">info:eu-repo/semantics/openAccess</subfield>
</datafield>
<datafield ind1="0" ind2=" " tag="506">
<subfield code="u">
http://inspire.ec.europa.eu/metadata-codelist/LimitationsOnPublicAccess/
     noLimitations
</subfield>
<subfield code="a">Access copy available to the general public</subfield>
<subfield code="f">Unrestricted</subfield>
</datafield>
<datafield ind1="4" ind2=" " tag="856">
<subfield code="n">trafair:prediction_gral_nox</subfield>
<subfield code="u">
```

```
http://atila.unizar.es:8081/geoserver/ows?service=wms&request=
    GetCapabilities
</subfield>
<subfield code="y">WMS</subfield>
</datafield>
<datafield ind1=" " ind2=" " tag="540">
<subfield code="u">https://creativecommons.org/licenses/by-nc/4.0/</subfield
    >
<subfield code="2">cc</subfield>
<subfield code="a">by-nc</subfield>
<subfield code="b">Creative Commons</subfield>
<subfield code="c">4.0</subfield>
<subfield code="f">CC BY-NC 4.0</subfield>
<subfield code="9">info:eu-repo/semantics/openAccess</subfield>
</datafield>
<datafield ind1="0" ind2=" " tag="506">
<subfield code="u">
http://inspire.ec.europa.eu/metadata-codelist/LimitationsOnPublicAccess/
    noLimitations
</subfield>
<subfield code="a">Access copy available to the general public</subfield>
<subfield code="f">Unrestricted</subfield>
</datafield>
<datafield ind1=" " ind2=" " tag="041">
<subfield code="a">
http://publications.europa.eu/resource/authority/language/ENG
</subfield>
</datafield>
<datafield ind1=" " ind2=" " tag="518">
<subfield code="a">22/01/2020-22/01/2020</subfield>
</datafield>
<datafield ind1=" " ind2=" " tag="654">
<subfield code="2">inspire</subfield>
<subfield code="a">http://inspire.ec.europa.eu/theme/ef</subfield>
<subfield code="a">
http://inspire.ec.europa.eu/metadata-codelist/TopicCategory/environment
</subfield>
<subfield code="a">http://inspire.ec.europa.eu/theme/ac</subfield>
<subfield code="a">
http://inspire.ec.europa.eu/metadata-codelist/TopicCategory/
    climatologyMeteorologyAtmosphere
</subfield>
</datafield>
</record>
```

Código B.3: Extracto del XML MARC21 de un conjunto de datos subido a Zaguan

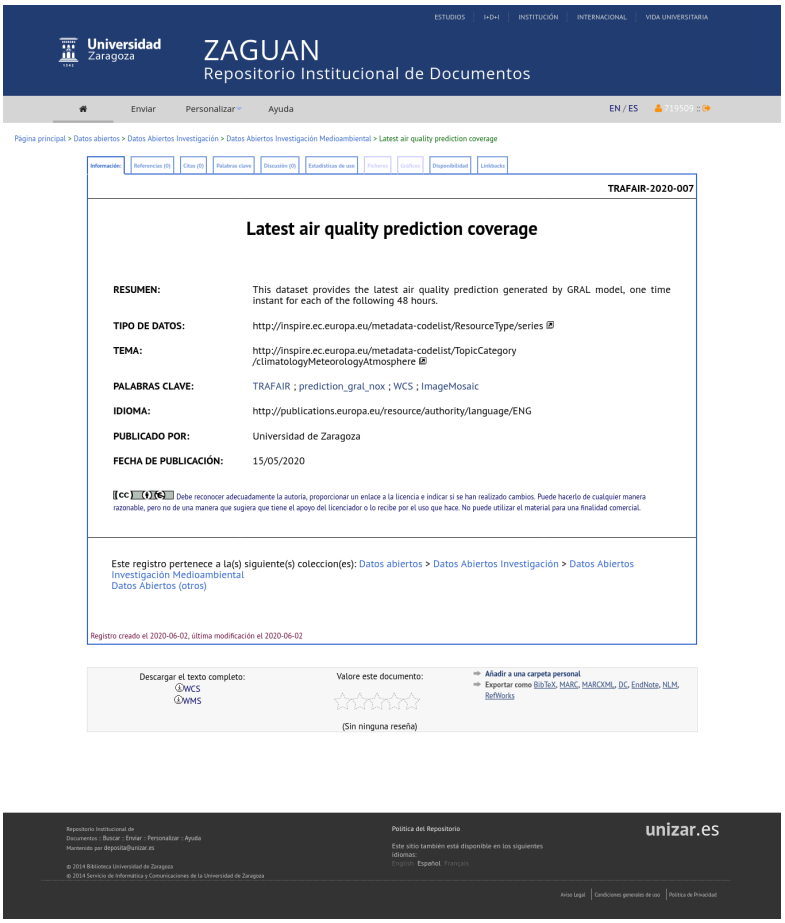


Figura B.3: Detalle de un conjunto de datos subido en el repositorio Zaguan

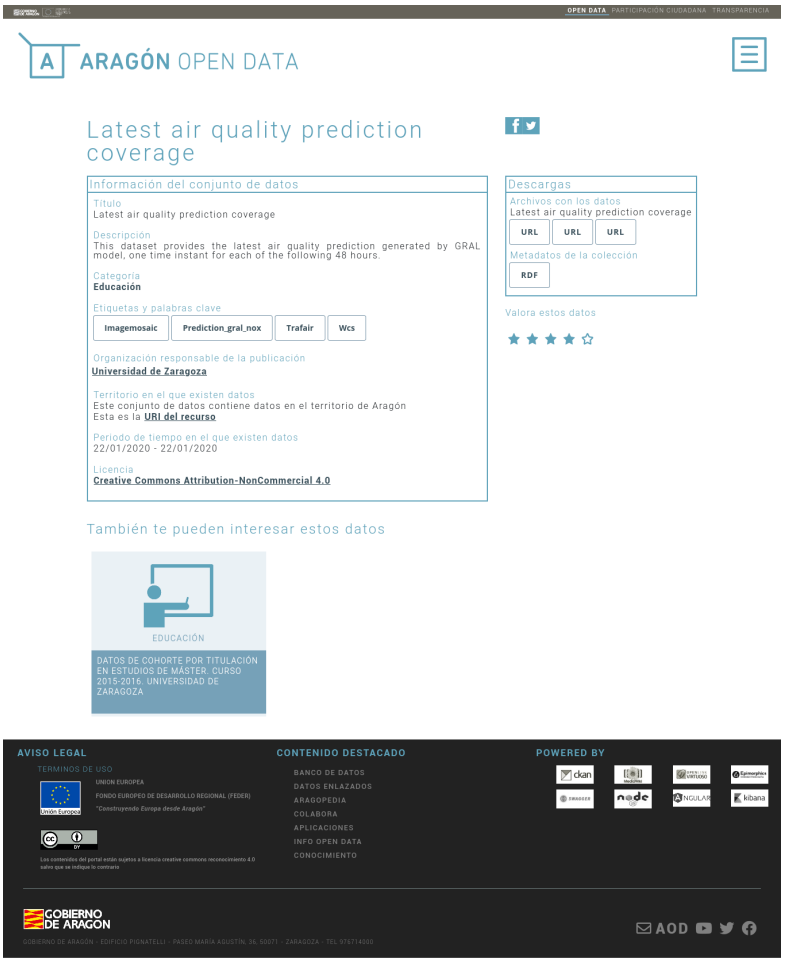


Figura B.4: Detalle de un conjunto de datos subido en el portal Aragón Open Data

```
<rdf:RDF>
<dcat:Dataset rdf:about="https://opendata.aragon.es/datos/catalogo/dataset/
    oai-zaguan-unizar-es-89320">
<dct:identifier>oai-zaguan-unizar-es-89320</dct:identifier>
<dct:spatial>
<rdf:Description rdf:nodeID="N1ef3179cd87249e4b03e143b447ed800">
<dct:title xml:lang="es">aragon</dct:title>
<ns2:Aragopedia.htmlComunidadAutonoma xml:lang="es">aragon2</ns2:Aragopedia.
    htmlComunidadAutonoma>
<rdf:resource>
http://opendata.aragon.es/recurso/territorio/ComunidadAutonoma/Aragon?
    api_key=e103dc13eb276ad734e680f5855f20c6
</rdf:resource>
</rdf:Description>
</dct:spatial>
<dcat:Distribution>
<dcat:Distribution rdf:about="https://opendata.aragon.es/dataset/oai-zaguan-
    unizar-es-89320/resource/ff83ae35-b6e3-4047-afd5-c52ef7ef687b">
<dct:title>Latest air quality prediction coverage</dct:title>
<dcat:accessURL rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
http://atila.unizar.es:8081/geoserver/ows?service=wms&request=
    GetCapabilities
</dcat:accessURL>
<dcat:accessURL rdf:resource="http://atila.unizar.es:8081/geoserver/ows?
    service=wms&request=GetCapabilities"/>
```

```
<dct:format>
<rdf:Description rdf:nodeID="Ndfc4cb343720454cbc7d1e61c02da91d">
<dct:MediaType>
<rdf:Description rdf:nodeID="N5e43d45212b64959bf8a72c0aefd1ae0">
<rdfs:value>None</rdfs:value>
<rdfs:label>URL</rdfs:label>
</rdf:Description>
</dct:MediaType>
</rdf:Description>
</dct:format>
<dct:format>URL</dct:format>
<dct:title xml:lang="es">Latest air quality prediction coverage</dct:title>
<dct:description xml:lang="es"/>
<dct:identifier rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
https://opendata.aragon.es/dataset/oai-zaguan-unizar-es-89320/resource/
     ff83ae35-b6e3-4047-afd5-c52ef7ef687b
</dct:identifier>
</dcat:Distribution>
</dcat:Distribution>
<dcat:keyword>Prediction_gral_nox</dcat:keyword>
<dcat:distribution>
<dcat:Distribution rdf:about="https://opendata.aragon.es/dataset/oai-zaguan-
     unizar-es-89320/resource/c755f1dc-a844-4e9f-8e39-07ae07dfbe6f">
<dct:title xml:lang="es">Latest air quality prediction coverage</dct:title>
<dct:identifier rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
https://opendata.aragon.es/dataset/oai-zaguan-unizar-es-89320/resource/
     c755f1dc-a844-4e9f-8e39-07ae07dfbe6f
</dct:identifier>
<dct:format>
<rdf:Description rdf:nodeID="Nc0c2e40824044c99abe5976628bc5f76">
<dct:MediaType>
<rdf:Description rdf:nodeID="Nc5fe3d24148a4fe2ba58455b348cfad3">
<rdfs:label>URL</rdfs:label>
<rdfs:value>None</rdfs:value>
</rdf:Description>
</dct:MediaType>
</rdf:Description>
</dct:format>
<dcat:accessURL rdf:resource="http://atila.unizar.es:8081/geoserver/ows?
     service=WCS&request=GetCapabilities"/>
<dct:format>URL</dct:format>
<dct:description xml:lang="es"/>
<dct:title>Latest air quality prediction coverage</dct:title>
<dcat:accessURL rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
http://atila.unizar.es:8081/geoserver/ows?service=WCS&request=
     GetCapabilities
</dcat:accessURL>
</dcat:Distribution>
</dcat:distribution>
<dct:title xml:lang="es">Latest air quality prediction coverage</dct:title>
<dcat:keyword xml:lang="es">Prediction_gral_nox</dcat:keyword>
<dcat:keyword>Trafair</dcat:keyword>
<dcat:contactPoint>
<vcard:Organization rdf:nodeID="N5a79a25fff074c0dbc3357a6f36380ec">
<vcard:fn>Universidad de Zaragoza (15/05/2020)</vcard:fn>
</vcard:Organization>
</dcat:contactPoint>
<dct:title>Latest air quality prediction coverage</dct:title>
<dct:identifier rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
https://opendata.aragon.es/datos/catalogo/dataset/oai-zaguan-unizar-es-89320
</dct:identifier>
<dcat:Distribution>
<dcat:Distribution rdf:about="https://opendata.aragon.es/dataset/oai-zaguan-
     unizar-es-89320/resource/57473188-2a7f-42ec-b57f-685b66182ec1">
<dct:description xml:lang="es"/>
<dct:title xml:lang="es">Latest air quality prediction coverage</dct:title>
<dct:format>URL</dct:format>
<dcat:accessURL rdf:resource="http://zaguan.unizar.es/record/89320"/>
<dct:title>Latest air quality prediction coverage</dct:title>
<dcat:accessURL rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http
     ://zaguan.unizar.es/record/89320</dcat:accessURL>
<dct:format>
<rdf:Description rdf:nodeID="Nd028358d108641a09829ddf3b8b9af90">
<dct:MediaType>
<rdf:Description rdf:nodeID="N2db2982df7e44d75a10f6d7079fb28b7">
<rdfs:label>URL</rdfs:label>
<rdfs:value>None</rdfs:value>
</rdf:Description>
</dct:MediaType>
</rdf:Description>
</dct:format>
<dct:identifier rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
https://opendata.aragon.es/dataset/oai-zaguan-unizar-es-89320/resource
```

```
     /57473188-2a7f-42ec-b57f-685b66182ec1
96 </dct:identifier>
97 </dcat:Distribution>
98 </dcat:Distribution>
99 <dcat:keyword xml:lang="es">Wcs</dcat:keyword>
100 <dct:issued rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime
     ">2020-06-06T01:10:12.034208</dct:issued>
101 <dcat:Distribution rdf:resource="https://opendata.aragon.es/dataset/oai-
     zaguan-unizar-es-89320/resource/c755f1dc-a844-4e9f-8e39-07ae07dfbe6f"/>
102 <dct:publisher rdf:resource="https://opendata.aragon.es/datos/catalogo/
     publicadores/universidad-de-zaragoza"/>
103 <dcat:distribution rdf:resource="https://opendata.aragon.es/dataset/oai-
     zaguan-unizar-es-89320/resource/ff83ae35-b6e3-4047-afd5-c52ef7ef687b"/>
104 <dct:license rdf:resource="https://creativecommons.org/licenses/by-nc
     /4.0/"/>
105 <dct:temporal>
106 <rdf:Description rdf:nodeID="N0d3e518273e74fcc85b232a126989292">
107 <ns1:timeInterval>
108 <dct:PeriodOfTime rdf:nodeID="Nd24a37201523490299c816d7419c9643">
109 <ns1:timehasEnd>
110 <rdf:Description rdf:nodeID="N21d5f849292049acba599e44cb81e498">
111 <ns1:timeInstant>
112 <rdf:Description rdf:nodeID="Nc2ad35a48447437e9b9568391e71be05">
113 <ns1:timeinXSDDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date
     ">2201-01-01</ns1:timeinXSDDate>
114 </rdf:Description>
115 </ns1:timeInstant>
116 </rdf:Description>
117 </ns1:timehasEnd>
118 <ns1:timehasBeginning>
119 <rdf:Description rdf:nodeID="Na834745869f248b4944d02d63060d643">
120 <ns1:timeInstant>
121 <rdf:Description rdf:nodeID="Nbb22c8c45d3341cfbd238d3bb14bd962">
122 <ns1:timeinXSDDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date
     ">2201-01-01</ns1:timeinXSDDate>
123 </rdf:Description>
124 </ns1:timeInstant>
125 </rdf:Description>
126 </ns1:timehasBeginning>
127 </dct:PeriodOfTime>
128 </ns1:timeInterval>
129 </rdf:Description>
130 </dct:temporal>
131 <dcat:keyword>Imagemosaic</dcat:keyword>
132 <dcat:keyword xml:lang="es">Imagemosaic</dcat:keyword>
133 <dcat:keyword xml:lang="es">Trafair</dcat:keyword>
134 <dcat:keyword>Wcs</dcat:keyword>
135 <dct:description>
136 This dataset provides the latest air quality prediction generated by GRAL
     model, one time instant for each of the following 48 hours.
137 </dct:description>
138 <dcat:distribution rdf:resource="https://opendata.aragon.es/dataset/oai-
     zaguan-unizar-es-89320/resource/57473188-2a7f-42ec-b57f-685b66182ec1"/>
139 <dct:modified rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime
     ">2020-06-06T01:10:12.034217</dct:modified>
140 <dct:publisher>
141 <foaf:Organization rdf:about="https://opendata.aragon.es/datos/catalogo/
     publicadores/b6ad71a9-de71-476e-a920-cbc4b9ea06a1">
142 <foaf:name>Universidad de Zaragoza</foaf:name>
143 </foaf:Organization>
144 </dct:publisher>
145 <dcat:theme>Educación</dcat:theme>
146 <dct:description xml:lang="es">
147 This dataset provides the latest air quality prediction generated by GRAL
     model, one time instant for each of the following 48 hours.
148 </dct:description>
149 </dcat:Dataset>
150 </rdf:RDF>
```

Código B.4: GeoDCAT-AP RDF de un conjunto de datos servido desde Aragón Open Data

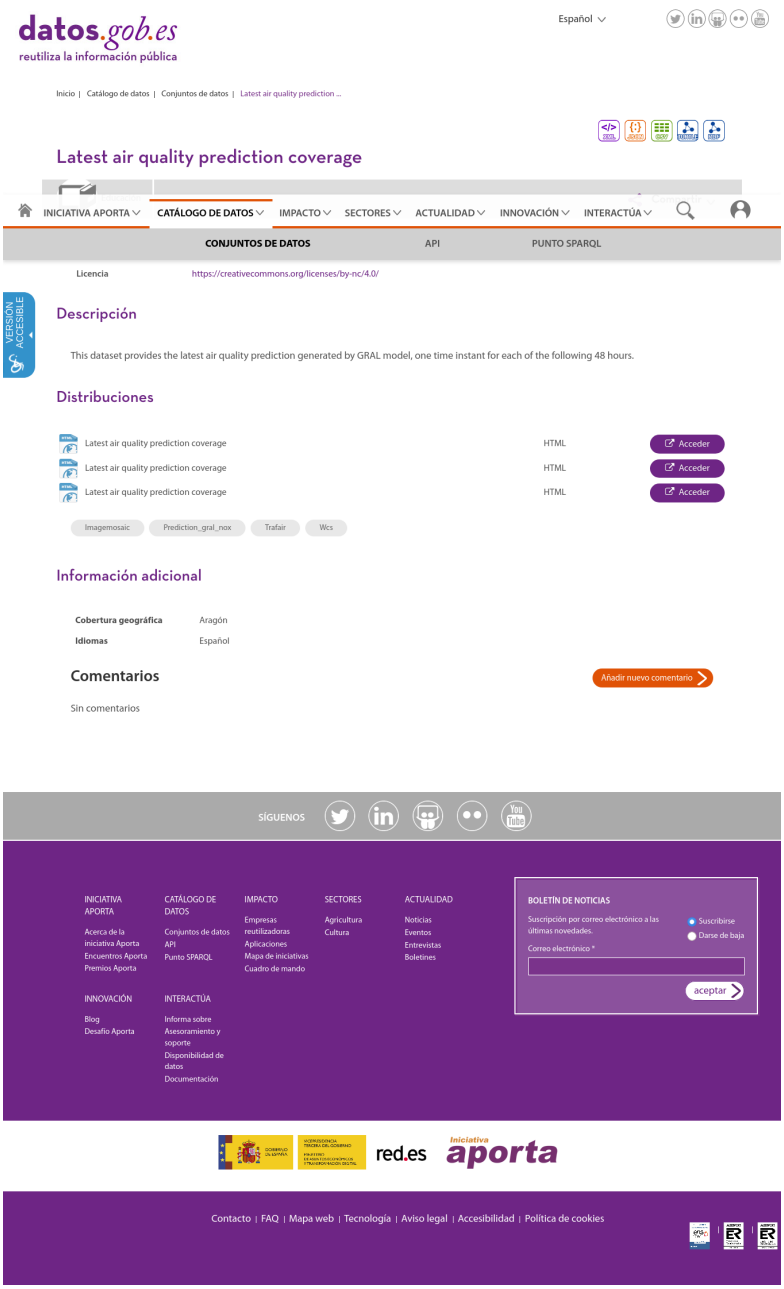


Figura B.5: Detalle de un conjunto de datos subido en el portal datos.gob.es

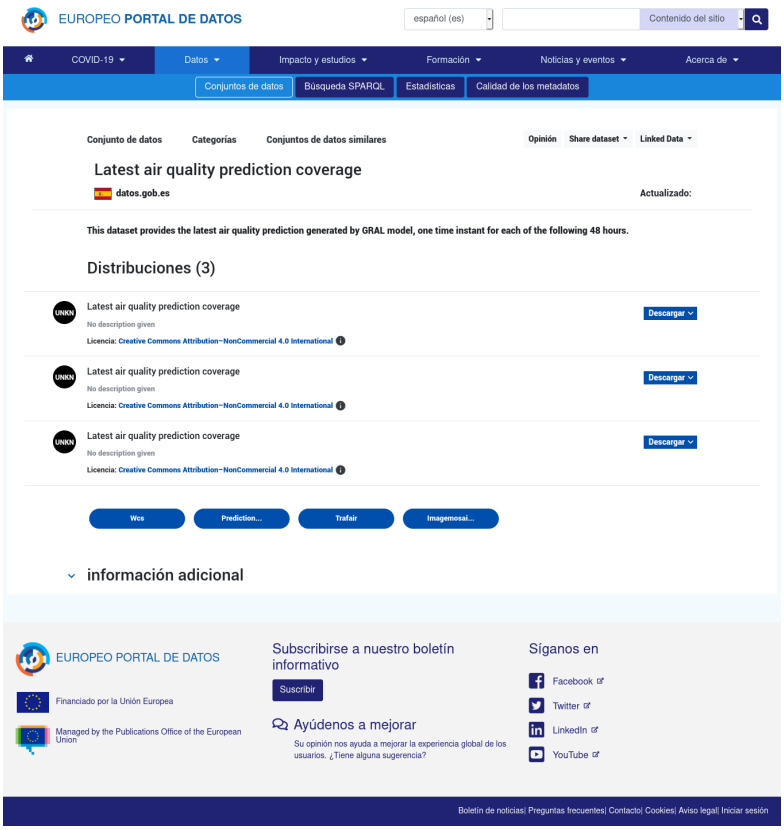


Figura B.6: Detalle de un conjunto de datos subido en el Portal Europeo de Datos